はじめに

このリファレンスマニュアルでは"EZ-BASIC"で使用できる命令、関数および、関連する用語について説明しています。
※その他のツールについての解説は、"User's Manual"を参照してください。

目次

○本書について …… 2
○基本命令 …… 3
○直接命令 …… 25
○拡張命令 …… 32
○2Dグラフィックス関連 …… 43
○3Dグラフィックス関連 …… 61
○サウンド …… 82
○文字列関数 …… 86
○数値関数 …… 92
○拡張関数 …… 101
○索引 …… 106

図表目次

○図1　変数 …… 3
○図2　配列 …… 6
○図3　多次元配列 …… 6
○図4　スクリーン構成 …… 32
○図5　スクリーン座標系 …… 43
○図6　Zバッファ方式 …… 44
○図7　Zバッファ問題 …… 44
○図8　Zソート …… 45
○図9　パレットとオフセット値 …… 51
○図10 テクスチャー座標系 …… 54
○図11 モデルの表示 …… 61
○図12 親子モデル …… 62
○図13 フォグの設定 …… 64
○図14 CAMROT …… 67
○図15 CAMVEC …… 68
○図16 FOGSET …… 69
○図17 LIGVEC …… 75
○図18 MODROT …… 80
○図19 コントローラのボタン番号 …… 104
○表1　JISコード表 …… 86
○表2　キーコード表 …… 102
○表3　コントローラのボタン対応 …… 104

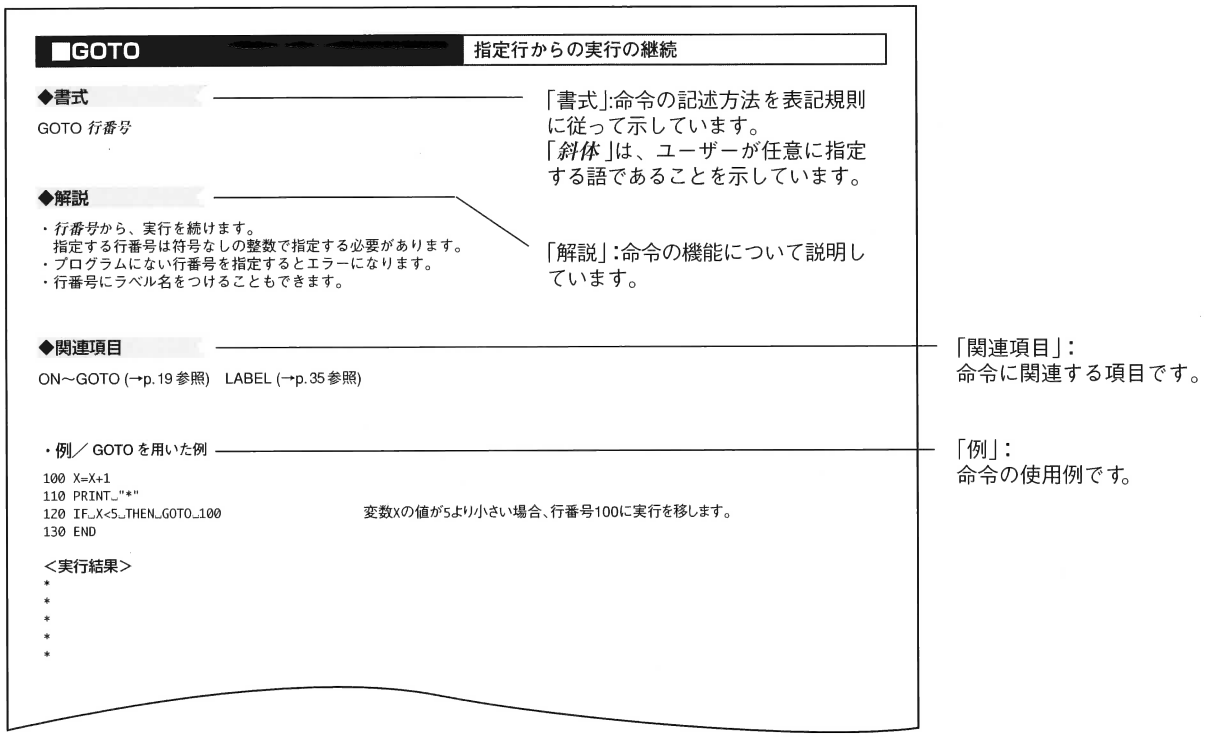
※本書は再発行いたしませんので、紛失しないよう大切に保管してください。

本書について

■構成

本書は、命令の機能ごとに以下の項目に分類してあります。
・基本命令…制御構造や判定構造など、BASICにおける基本的な命令です。
・直接命令…プログラムの編集やファイルの操作など、直接的な機能を持つ命令です。
・拡張命令…画面の分割表示やコントローラの振動など、EZ-BASIC特有の拡張された機能を扱う命令です。
・2Dグラフィックス関連…背景やプリミティブなどのグラフィックスを扱う命令です。
・3Dグラフィックス関連…モデルやカメラ、ライトなどを扱う命令です。
・サウンド…BGMや効果音(SE:Sound Effect)などのサウンドに関する命令です。
・文字列関数…文字列の処理に関する関数です。
・数値関数…数学関数や数値を処理する関数です。
・拡張関数…3Dオブジェクトの当たり判定やキーボードからの入力など、EZ-BASIC特有の拡張された機能を扱う関数です。

■リファレンスの見方



※例で示しているプログラムを入力して実行する場合、モデルの設定などを初期状態にするためにINIT命令(→p.34参照)
およびNEW命令(→p.28参照)を実行してください。

■表記規則

斜体 ――― プログラムを書く際、ユーザーが任意の名称を定義する項目を示しています。
START+SELECT ――― STARTボタンとSELECTボタンを同時に押すことを示しています。
{FIRST|LAST} ――― FIRSTとLASTの中からひとつを選ぶことを示しています。
[ABC] ――― ABCは省略可能であることを示しています。
ABC... ――― ABCが0回以上繰り返すことを示しています。
(一連の行) ――― 0行以上のプログラムの繰り返しを示しています。

" ␣ "は半角のスペースを挿入することを意味しています。

基本命令

■プログラムの記述

◆記述に関する仕様

・プログラムは、行番号からはじまる命令文の含まれる行で構成されます。
・行番号として扱える範囲は、1～999999999までの整数です。0および負数は用いることができません。
・プログラムは行番号の小さい順に並び替えられ、行番号の小さい順に実行されます。
・複数の行が同一の行番号を持つことはできません。同一の行番号を書いた場合、最新の行がその行番号に割り当てられます。
・プログラムは、最大20000行まで記述することができます。
・1行につき、行番号を含めて最大132文字まで記述できます。
・行番号には、ゼロ列を先行して書くことができます。例えば行番号10は、00010のように記述することができます。
・変数名、命令語、副プログラム名をはじめとする機能語は英大文字と英小文字を区別しません。
・JIS規格のFullBASICおよび他社製BASICとの互換性はありません。

◆変数

・変数は、数値または文字列の値を保持する機能を持ちます。
・数値を保持する変数を数値変数、文字列を保持する変数を文字列変数と呼びます。

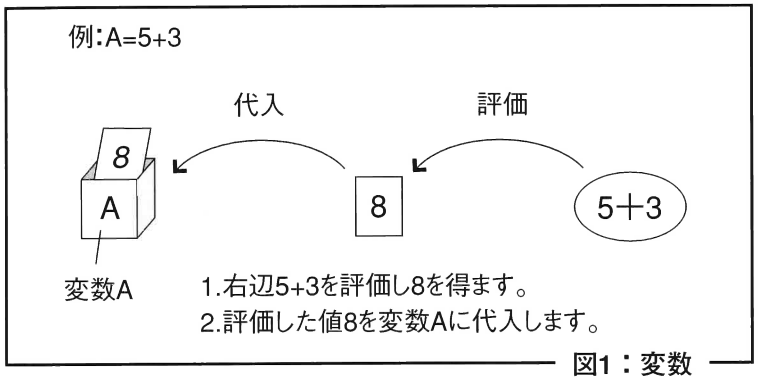


図1：変数

変数に値を入れることを「代入」と呼びます。
・例
A=5+3
この式は、右辺5+3を評価し、その値8を左辺の変数Aに代入することを表しています。

・数値変数および文字列変数に使用できる変数はあわせて1000個までです。
・変数名は、文字列変数に用いる$記号を含めて10文字までが認識されます。それを超す文字は無視されます。
・変数名には、英字、数字および下線記号(_)を用いることができます。ただし先頭に数字を用いることはできません。
・命令などの機能語として予約されている語は変数名として使用できません。

◆全角かな漢字

・EZ-BASICでは、日本語などの全角文字をプログラム内に使うことができます。
・全角文字は、1文字につき半角2文字の長さを持ちます。全角文字を削除すると、半角1文字分が削除され、残りの半角1文字分が空白に置き換えられます。
・コメントや文字列値としてのみ用いることができます。変数名、副プログラム名などの名前に用いることはできません。
・全角かな漢字の入力については“User's Manual”のかな漢字変換の項目を参照してください。

■演算子

◆算術演算子

数値計算を行うための演算子です。演算は一般的な代数に従います。

＋ ――― 加算、正符号
－ ――― 減算および符号反転、負符号
＊ ――― 乗算
／ ――― 除算。0で除算するとエラーになります。
＾ ――― べき乗。0^0は1になります。
e ――― 指数記号。3e2は3×10^{2}、5e−2は5×10^{-2}を表します。

・例

```
A=23+5*12-36/3^2     算術演算子の例。値79が変数Aに代入されます。
A=-A                 符号反転の例。変数Aの符号を反転します。
A=3e+2               指数表現による数値表現。3.0*10^2 を表します。
```

◆比較演算子

演算子をはさむ左辺と右辺を比較し、その構文が真（比較演算が合う）または偽（比較演算が合わない）を評価します。

= ――――― 左辺値と右辺値が等しい。
> ――――― 左辺値は右辺値より大きい。
< ――――― 左辺値は右辺値より小さい。
>=,=> ――― 左辺値は右辺値と比べて等しいか大きい。
<=,=< ――― 左辺値は右辺値と比べて等しいか小さい。
<>,>< ――― 左辺値と右辺値は等しくない。

◆論理演算子

演算子をはさむ左辺と右辺を比較し、その構文の真偽を評価します。

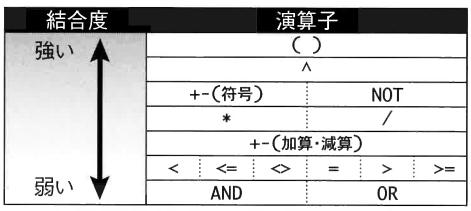
AND ――――― 左辺の評価値が真で、かつ右辺の評価値も真である。（論理積）
OR ――――― 左辺の評価値が真または、右辺の評価値が真である。（論理和）
NOT ――――― 右辺の評価値を否定。（論理否定）

・例

```
IF A<B␣AND␣C=D␣THEN␣PRINT"HAPPY"                A<BかつC=Dの場合、THEN句が実行されます。
IF NOT␣A=B␣THEN␣PRINT"UNHAPPY"                  A=Bでない場合、THEN句が実行されます。
IF A=B␣OR␣C<D␣AND␣C=10␣THEN␣PRINT"HAPPY?"      A=Bまたは、C<DかつC=10である場合、THEN句が実行されます。
IF (A=B␣OR␣C<D)␣AND␣C=10␣THEN␣PRINT"HAPPY!"    A=BまたはC<Dで、かつC=10である場合、THEN句が実行されます。
```

◆演算順序および結合度

・式は通常、左から右に評価されます。演算子の結合度合またはカッコによる優先によって評価順序が決定されます。
・カッコ()ではさむことで、演算を優先的に行うことができます。
・演算子の結合度
　結合度が強いほど評価が優先して行われます。



・例

```
IF␣NOT(A*3-4<=B)AND((-A+B>C^3)OR(B=C))␣THEN␣250
```

これは以下の順序で評価を行います。
まず、結合度に従って算述演算式を評価した後、各比較演算式を評価します。
（式1）:A*3-4<=B　　A*3-4 の値がBの値より等しいか小さい。
（式2）:-A+B>C^3　　-A+B の値がC^3 の値より大きい。
（式3）:B=C　　Bの値とCの値が等しい。
次に各式の結果から、論理演算を行います。

```
IF NOT(式1) AND ((式2)OR(式3))THEN 250
```

（式1）が真ではなく、かつ（式2）が真または（式3）が真である場合、THEN 句が実行されます。

■数値

◆数値の表現

・数値を表す文字として数字(0～9)、小数点記号(.)、符号(+-)、指数記号(eまたはE)を用いることができます。
・数値は以下の形式で表現されます。
　(1)小数点なし指数部なし形式。500、-2300 など
　(2)小数点あり指数部なし形式。12.34、.45 など
　(3)小数点あり指数部あり形式。-2.9090e+20など
　(4)小数点なし指数部あり形式。1e+14、-5e-30など
　桁数が十分大きい場合、(3)または(4)による形式で表現されます。
・小数点あり形式で表現される数値において整数部がゼロである場合、そのゼロを省略できます。例えば-0.5は-.5として書くことができます。
・数値は32ビット浮動小数点2進数方式で保持されます。有効桁数は10進数で6桁程度の精度を持ちます。
十分大きい数値に、十分小さい数値を加える場合、有効桁数により十分小さい数値が0に丸められる可能性があります。
小数を扱う場合、2進数による表現のため有効桁数内であっても誤差が生じる場合があります。
・数値として扱える範囲は、-3.40282e+38～3.40282e+38の範囲です。
指数部に扱える範囲はおよそe-37～e+38の範囲です。

◆数値変数

・数値のみの代入ができます。文字列など数値でない値を代入しようとするとエラーになります。
・保持できる値は、-3.40282e+38～3.40282e+38です。

■文字列

◆文字列

・文字列には、英字、数字、記号、全角かな漢字を使用することができます。
・前後にダブルクォーテーション(")を書くことにより、文字列とみなされます。

◆文字列変数

・変数名の末尾に$記号を加えることで、文字列変数として解釈されます。
・数値など、文字列以外の値を代入しようとするとエラーになります。
・文字列変数に代入できる文字列は最大512文字です。ただし、全角かな漢字は2文字分として評価されます。

◆文字列の連結

・文字列の連結を行う記号(+または&)により文字列どうしを連結した値を得ることができます。
A$="ABC"+"DEF"&"GHI"は、値"ABCDEFGHI"を文字列変数A$に代入します。
文字列の連結に用いる記号+および&は、同じ機能を持ちます。

◆部分文字列指定

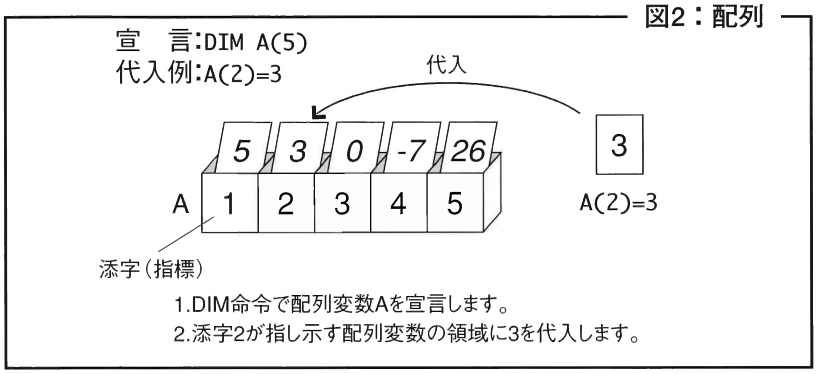
・部分文字列指定を用いて、変数に格納されている文字列の一部を指定することができます。
・文字列変数A$に対し、書式A$(m:n)を用いることで文字列変数A$に格納されている値のm文字からn文字目までの文字列を指し示すことができます。
A$="BASIC STUDIO"である場合、A$(8:11)は文字列"TUDI"を表します。
・部分文字列指定した範囲に文字列を代入することができます。
代入する文字列と、部分文字列指定した文字列は置き換えられます。
よって、代入する文字列が部分文字列指定した部分より短い場合、文字列全体が短くなり、代入する文字列が部分文字列指定した部分より長い場合、文字列全体が長くなります。
A$="BASIC"の時、A$(1:4)="AB"を実行すると部分文字列指定A$(1:4)が指す値"BASI"を代入する文字列"AB"で置き換え、結果として文字列"ABC"を得ます。
・部分文字列指定A$(m:n)で、m>nの時、m文字目の直前に文字列を挿入できます。
A$="SING"の時、A$(2:1)="TR"を実行すると部分文字列指定A$(2:1)の表す2文字目の直前に文字列"TR"を挿入し、結果として文字列"STRING"を得ます。

■配列

◆配列の使用

・配列は数値または文字列をまとめて格納して扱うことができます。
　配列は、数値配列または文字列配列に区別されます。
・DIM命令(→p.11参照)により配列を宣言することで、配列を使用できるようになります。
・配列に格納された各要素は、**添字**または**指標**と呼ばれる数値によって指定することができます。
・配列の添字が小数点以下の数字を含む場合、小数点以下を四捨五入し整数として扱います。

図2：配列



・数値配列は、配列の要素に数値のみ格納することができます。それ以外の値を代入しようとするとエラーになります。
・数値配列ひとつが格納できる要素数は最大1000000個までです。
・文字列配列は配列の要素に文字列のみ格納することができます。それ以外の値を代入しようとするとエラーになります。
・文字列配列が格納できる最大要素数は、空きメモリ容量によって決定されます。

◆多次元配列

・配列は1次元配列に加え、2次元配列、3次元配列を宣言することができます。
・2次元および3次元の多次元配列を宣言するには、DIM命令に続く配列要素をコンマ（,）で区切って書きます。

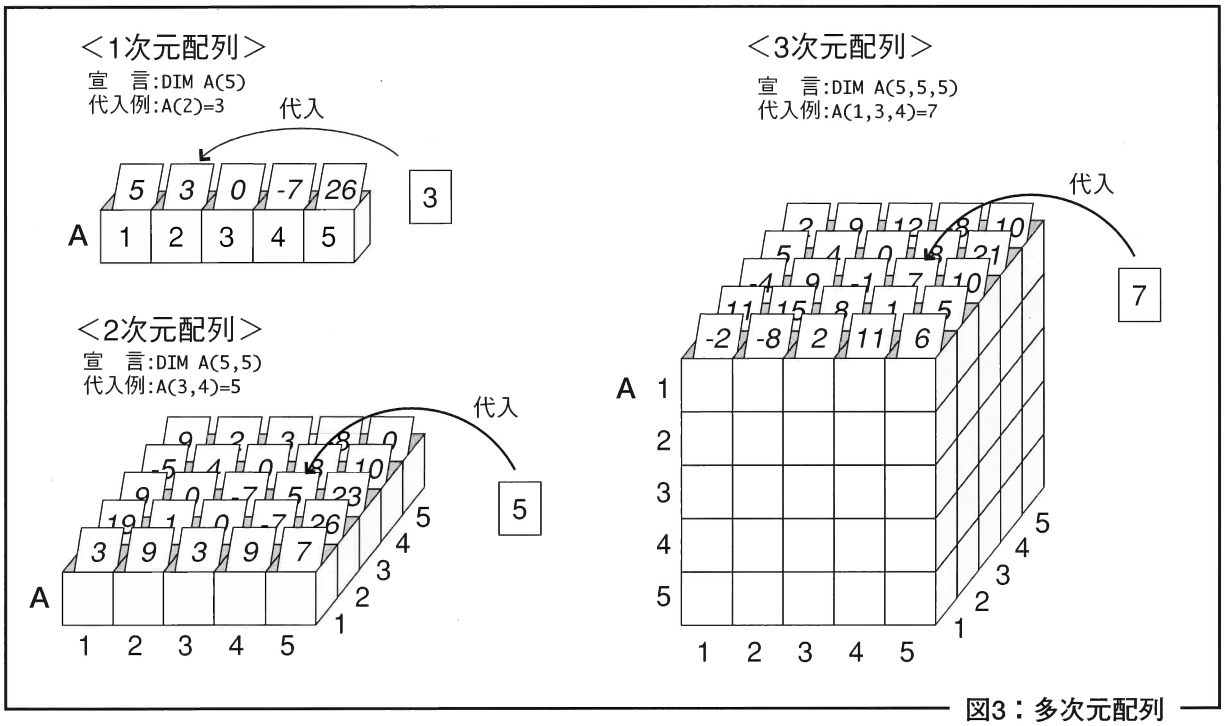


図3：多次元配列

■副プログラム

◆副プログラムの機能

・副プログラムはある一定の機能をもった命令の集まりをまとめ、その副プログラムを呼び出すことで、まとめられた命令を実行することのできる機能です。
・副プログラムはユーザーが自由に定義できます。
・副プログラムは最大1000種類まで定義できます。

◆引数

・副プログラムには**引数**として、数値、文字列や変数を渡すことができます。
副プログラムに対して引数を渡すことで、それをもとに処理をさせることができます。

・例

```
100 DO
110  CALL␣CONTR(1)          副プログラムCONTRに、引数として数値1を渡して実行します。
120  CALL␣CONTR(2)          副プログラムCONTRに、引数として数値2を渡して実行します。
130  FSYNC␣1
140 LOOP
150 SUB␣CONTR(P)            副プログラムCONTR定義部分
160  TILEDISP␣P             引数Pで受取った数値のタイルプリミティブを表示します。
170  X=PAD(P,1)-PAD(P,2)    引数P側のコントローラ端子に接続されたコントローラの入力を調べます。
180  TILEPOS␣P,X+320,40,X+360,60,0   タイルプリミティブの配置場所を更新します。
190 END␣SUB                  副プログラムの定義終了
```

…副プログラムCONTRを、150行～190行に定義します。
…この副プログラムは、タイルプリミティブ(画面上の白い四角形)をコントローラの方向キー左・右によって移動させる処理をします。
…この副プログラムは、変数Pという引数をひとつ使用します。変数Pをもとに、副プログラムはコントローラ番号やタイルプリミティブの位置を制御します。
変数Pには110行～120行で数値の1または2が渡され、それによりコントローラ端子1に接続されたコントローラの処理とコントローラ端子2に接続されたコントローラの処理を行います。

・定義用に仮に使用する引数を、**仮引数**と呼びます。これに対し実行時の引数を、**実引数**と呼びます。
・仮引数と実引数の型（数値や文字列といった種類）および引数の個数は同じでなければなりません。

◆配列の引数

・副プログラム定義内の仮引数に、配列を用いることができます。これを**仮配列宣言**と呼びます。
仮配列宣言の書式は、仮配列変数名に続けてカッコを続けることで行います。多次元配列の場合は、そのカッコ内に（次元数-1）だけのコンマ(,)を書きます。
・仮配列宣言での書式

```
SUB␣SP1(CH$())      CH$()は1次元の文字列配列の仮配列宣言
SUB␣SP2(AR(,,))     AR(,,)は3次元の数値配列の仮配列宣言
```

・引数として渡す際の書式
プログラム実行時、副プログラムに配列を渡す際は配列名のみを書きます。

```
CALL␣SP1(DM$)       1次元文字列配列DM$は上記副プログラムSP1に対応する実引数
CALL␣SP2(DM)        3次元数値配列DMは上記副プログラムSP2に対応する実引数
```

◆値と参照

・引数には、数値や文字列といった値を渡す場合と、変数を渡す場合があります。
・引数に変数名を渡した場合、引数はその変数への**参照**となります。
　参照により、仮引数への代入がすべて渡した変数へ反映されます。

・例

```
100 SUB␣SP1(A)                  引数の値を2倍にする副プログラム
110  A=A*2
120 END␣SUB
130 X,Y=5                       変数X,Yに5を代入
140 CALL␣SP1(X)                 変数Xを参照によって渡します。
150 PRINT␣X                     参照により変数Xへの代入が行われるため値が10になります。
160 CALL␣SP1((Y))               変数Yの値5を渡します。
170 PRINT␣Y                     値のみを渡すため、変数Yの値は影響ありません。
180 END
```

＜実行結果＞

```
10
5
```

・変数の参照を渡さず、変数の中身の値を渡したい場合、変数をカッコ( )で囲います。

◆再帰

・副プログラム内で、その副プログラム自身を呼び出すことができます。これを**再帰的な呼び出し**と呼びます。

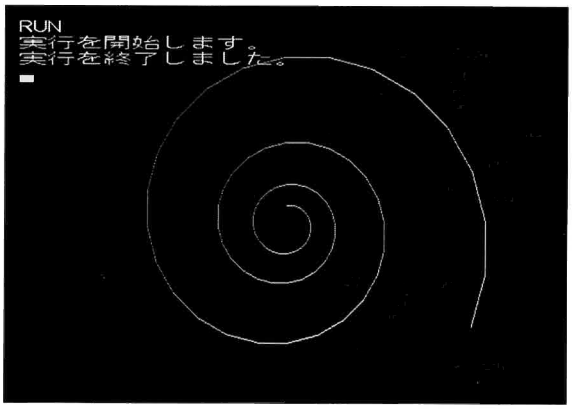
・例

```
100 CALL␣RECURSION(1,320,224,0,6)
110 SUB␣RECURSION(ID,X,Y,R,L)
120  RX=X+L*COS(R)
130  RY=Y+L*SIN(R)
140  RR=R+RAD(15)
150  RL=L*1.03
160  LINEDISP␣ID
170  LINECOL␣ID,Y/2,X/3,ID*3,128
180  LINEPOS␣ID,X,Y/2,RX,RY/2,0
190  IF␣ID<80␣THEN␣CALL␣RECURSION(ID+1,(RX),(RY),(RR),(RL))
200 END␣SUB
210 END
```

＜解説＞

100　副プログラムRECURSION を呼び出します。
110　プリミティブ番号ID、ライン始点(X,Y)、ライン終点方向R、ラインの長さLを引数に得ます。
120　引数からライン終点(RX,RY)を求めます。
140　再帰的に呼ぶRECURSIONに渡す、次のプリミティブの角度を計算します。
150　再帰的に呼ぶRECURSIONに渡す、次のプリミティブの長さを計算します。
190　プリミティブ番号が80より小さい場合、RECURSION 自身を新しい引数を渡して呼び出します。
　　80に等しいか大きい場合、呼び出しは行わず副プログラムを終了します。

＜実行結果＞



■CALL　副プログラムの呼び出し

◆書式

CALL *副プログラム名* [(*第1引数,第2引数...*)]

◆解説

・SUB～END SUB(→p.10参照)命令で定義した副プログラムの内容を呼び出して実行します。
・副プログラムに対し、引数を与えて実行することができます。引数はカッコ内に書き、2つ以上の引数がある場合はコンマ(,)で区切り列挙します。(→例2)
・SUB～END SUB命令によって定義した副プログラムの引数とCALL命令で呼び出す副プログラムの引数の数および引数の型が一致していないとエラーになります。
・呼び出した副プログラムの実行が終了すると、CALL命令の次の行から実行が継続されます。
・引数に変数名を指定すると、引数が参照によって副プログラムへ渡されます。(→例3)
・副プログラム内からその副プログラム自身を呼び出す、再帰的な呼び出しを行うことができます。

◆関連項目

SUB～END SUB (→p.10参照)

・例1／副プログラムを用いた例

```
100 CALL␣SUBP1              副プログラムSUBP1を呼び出して実行します
110 END                     100行でプログラムを実行した後、この行で終了します
120 SUB␣SUBP1               副プログラム定義の開始行
130  PRINT␣"TEST"
140 END␣SUB                 副プログラム定義の終了行
```

・例2／引数を持つ副プログラム

```
100 CALL␣SUBP2(3,5,9)       引数に3,5,9を渡して副プログラムSUBP2を実行します。
110 CALL␣SUBP2(23,0,9)      引数に23,0,9を渡して副プログラムSUBP2を実行します。
120 END                     110行実行後、この行で終了
130 SUB␣SUBP2(A,B,C)        副プログラム定義の開始行
140  PRINT␣A+B+C            引数A,B,Cの和を表示します
150 END␣SUB                 副プログラム定義の終了行
```

＜実行結果＞

```
17
32
```

・例3／参照を用いた副プログラム

```
100 A=5
110 B=5
120 CALL␣SUBP3(A,(B))       変数Aは参照によって、変数Bは値によって副プログラムに引数を渡します
130 PRINT␣A,B               変数A,Bが副プログラムSUBP3によってどう変化したか表示
140 END
150 SUB␣SUBP3(X,Y)          副プログラム定義の開始行
160  X=10                   仮引数Xに10を代入。実行時は参照により変数Aの値が10となります
170  Y=10                   仮引数Yに10を代入。値によって渡される変数Bには影響が及びません
180 END␣SUB                 副プログラム定義の終了行
```

＜実行結果＞

```
10   5
```

■SUB～END SUB/EXIT SUB　副プログラムの定義

◆書式

(1) 定義
SUB *副プログラム名* [(*第1引数,第2引数...*)]
　　(一連の行)
END SUB

(2) 副プログラムを出る
EXIT SUB

◆解説

・SUB行からEND SUB行までの一連の行を副プログラム(→p.7参照)として定義します。
・定義した副プログラムは、CALL命令(→p.9参照)に続けて副プログラム名を書くことで呼び出され、定義した一連の行を実行します。
・副プログラム名の後にカッコに続けて仮引数を書くことで、その副プログラムで引数を用いることができます。
複数の仮引数がある場合、コンマ(,)で区切って列挙します。
・副プログラムの引数として定義される変数は、その副プログラム内でのみ有効な**局所変数**として扱われます。(→例2)
副プログラム外部で定義される変数名と局所変数名が同一の場合、局所変数が優先して使用されます。
局所変数は副プログラムの実行が終了すると破棄されます。
・CALL命令によって副プログラムを実行する際、実引数が変数の場合、仮引数は実引数の参照として扱われます。
参照により、仮引数への代入は実引数へ反映されます。
・EXIT SUB命令を実行すると副プログラムの実行を終了し、CALL命令によって呼び出された行の次の行から実行を継続します。
EXIT SUB命令は副プログラムの外側に書くとエラーになります。
・プログラムの実行がSUB行に到達した場合、何も実行しないでEND SUB行の次の行から実行を継続します。

◆関連項目

CALL (→p.9参照)

・例1／副プログラムを用いた例

```
100 CALL␣SEISEKI("花乃",85,70)          引数"花乃",85,70を渡して副プログラムSEISEKIを実行
110 CALL␣SEISEKI("一郎太",23,40)        引数"一郎太",23,40を渡して副プログラムSEISEKI実行
120 SUB␣SEISEKI(NAME$,TEST1,TEST2)      副プログラム定義開始行
130  PRINT␣NAME$;"さんの成績"
140  PRINT␣TEST1;"点",TEST2;"点"
150  PRINT␣"平均",(TEST1+TEST2)/2;"点"
160 END␣SUB                             副プログラム定義終了行
170 END                                 110行終了後副プログラム定義部分は飛ばされ、この行を実行します
```

＜実行結果＞

```
花乃さんの成績
85 点　 70 点
平均　 77.5 点
一郎太さんの成績
23 点　 40 点
平均　 31.5 点
```

・例2／局所変数

```
100 SUB␣SP1(A)          この変数Aは局所変数として機能します。
110  A=A*2              局所変数Aへの代入となります。
120 END␣SUB             副プログラム終了後、局所変数Aは破棄されます。
130 A=4                 この変数Aは広域変数として機能します。
140 CALL␣SP1(5)
150 PRINT␣A             広域変数Aの内容が表示されます。
```

＜実行結果＞

```
4
```

■DIM　配列の宣言

◆書式

(1) 配列宣言
　DIM *配列名*(*最大値*)

(2) 下限値・上限値つき配列宣言
　DIM *配列名*(*下限値* TO *上限値*)

(3) 多次元配列宣言
　DIM *配列名*(*最大値,最大値...*)

(4) 複数の配列宣言
　DIM *配列名1,配列名2...*

◆解説

・配列を宣言します。配列を宣言することで、数値または文字列の集まりを扱うことができます。
・配列に格納された値は、添字(指標)と呼ばれる数値によって指し示すことができます。
・*配列名*に続けてカッコ内に*最大値*を指定することで、配列の要素をその最大値の数だけ確保します。(書式(1))
・*配列名*に続けてカッコ内に*下限値*および*上限値*を指定することで、*下限値*から*上限値*までの範囲の大きさを持つ配列を宣言します。添字(指標)はこの下限値から上限値までの範囲を指し示します。(書式(2))
・*最大値*をカッコ内にコンマ(,)で区切って記述することで多次元配列(→p.6参照)を宣言します。多次元配列は最大3次元配列まで宣言することができます。(書式(3))
　*最大値*を2つ書くことで2次元配列を、*最大値*を3つ書くことで3次元配列を宣言します。
・配列をコンマ(,)で区切って書くことで、複数の配列を宣言することができます。(書式(4))
・文字列配列を宣言する場合は、変数名の最後に"$"記号を加えます。
・下限値を指定していない場合、暗黙的に下限値が1となります。これはOPTION BASE命令(→p.19参照)によって0に変更することができます。

◆関連項目

OPTION BASE(→p.19参照)

・例1／配列宣言の例

```
DIM A(6)                    数値6つの大きさを持つ1次元の数値配列を宣言。添字の範囲は1～6
DIM B(10,5)                 2次元の数値配列を宣言
DIM B$(100)                 1次元の文字列配列を宣言
DIM C(-5 TO 5)              1次元の数値配列を宣言。添字の範囲は-5～5となる
DIM D(10,10,10),D$(50)      大きさ10×10×10の3次元数値配列と大きさ50の文字配列を連続して宣言します。
```

・例2／配列を用いた例

```
100 DIM␣TEST$(0␣TO␣2)       1次元の文字列配列TEST$を宣言。添字の範囲を0～2 に設定
110 TEST$(0)="JAPANESE:75"  配列TEST$の添字0が指す要素に文字列"JAPANESE:75"を代入
120 TEST$(1)="ENGLISH:50"   配列TEST$の添字1が指す要素に文字列"ENGLISH:50"を代入
130 TEST$(2)="PHYSICS:50"   配列TEST$の添字2が指す要素に文字列"PHYSICS:50"を代入
140 PRINT␣TEST$(0),TEST$(1),TEST$(2)
150 END
```

＜実行結果＞

```
JAPANESE:75   ENGLISH:50   PHYSICS:50
```

■DO～LOOP/EXIT DO　繰り返しの実行

◆書式

(1) 繰り返し
　DO ［{*WHILE*|*UNTIL*} *論理式*］
　　(一連の行)
　LOOP ［{*WHILE*|*UNTIL*} *論理式*］

(2) 繰り返しから出る
　EXIT DO

◆解説

・DOからLOOP間の一連の行を繰り返し実行します。
・DO行またはLOOP行に、繰り返しから出る条件となる*論理式*を書くことができます。
・DO行に条件を書くと、始めに繰り返しから出る条件を判定します。
　LOOP行に条件を書くと、少なくとも一度は繰り返しの一連の行を実行してから繰り返しから出る条件を判定します。
・WHILEに続けて*論理式*を記述すると、その論理式が偽のときに繰り返しから出ます。
・UNTILに続けて*論理式*を記述すると、その論理式が真のときに繰り返しから出ます。
・繰り返しを出ると、LOOP文の次の行から実行を継続します。
・EXIT DOが実行されると一連の行の実行を中断し、繰り返しを出ます。(→例2)

◆関連項目

FOR TO STEP～NEXT(→p.14参照)

・例1／DO～LOOPを用いた繰り返しの例

```
100 DO                                  繰り返し開始行
110  A=A+1
120  PRINT␣"HELLO"
130 LOOP␣UNTIL␣A>3                      変数Aの値が3より大きくなるまで繰り返しを継続
140 END
```

・例2／EXIT DOを用いて繰り返しを出る例

```
100 DO
110  A=A+10                             変数Aの値を繰り返しごとに10ずつ増やします。
120  PRINT␣A
130  IF␣A>50␣THEN␣EXIT␣DO              変数Aの値が50を超えると繰り返しを出ます。
140 LOOP
150 END
```

＜実行結果＞

```
10
20
30
40
50
60
```

・例3／WHILEを用いて繰り返しを出る例

```
100 DO␣WHILE␣A<10␣AND␣B<5              変数Aの値が10より小さく、かつ変数Bの値が5より小さい間は繰り返します。
110  A=A+2
120  B=B+1
130  PRINT␣A,B
140 LOOP
150 END
```

＜実行結果＞

```
2    1
4    2
6    3
8    4
10    5
```

■END　プログラムの実行終了

◆書式

END

◆解説

・プログラムの実行を終了します。

■FOR TO STEP～NEXT/EXIT FOR　繰り返しの実行

◆書式

(1)繰り返し
　FOR *制御変数名*= *初期値* TO *終了値* [STEP 増分]
　　(一連の行)
　NEXT *制御変数名*

(2)繰り返しから出る
　EXIT FOR

◆解説

・FOR文からNEXT文までの一連の行が含まれるFOR区を、繰り返し実行します。
・繰り返しは、*制御変数*の値が*初期値*から始まり、*終了値*を超えるまで行われます。
・NEXT文に到達すると*制御変数*に*増分*が加算され、FOR文に戻ります。このとき*制御変数*の値が*終了値*を超えている場合、FOR区を出てNEXT文の次の行から実行を継続します。
　*終了値*を超えていない場合、引き続きFOR区内の一連の行を繰り返します。
　NEXTに続ける*制御変数名*は、FOR文で用いている*制御変数名*と同じにしなければなりません。
・STEPに続けて*増分*の値を記述すると、NEXT文で*制御変数*に加算される値を設定することができます。(→例2)
　STEP句を省略をすると暗黙的に増分として1が設定されます。
・EXIT FOR文に到達すると、FOR区の繰り返しを出ます。EXIT FOR文はFOR区の中に記述しなければなりません。(書式(2))
・FOR区内に新たにFOR区を記述することで、内側のFOR区を外側のFOR区で繰り返し実行させる、入れ子構造にすることができます。(→例3)
　この場合、内側に含まれるFOR区で使用する制御変数名を、外側のFOR区と同じ制御変数名にすることはできません。
・制御変数に代入される始値が限界を超えている場合、FOR区の一連の行は実行されず、NEXT行の次の行から実行されます。

◆関連項目

DO～LOOP(→p.12参照)

・例1／FOR TO STEP～NEXTを用いた繰り返し文

```
100 SUM=0
110 FOR␣A=1␣TO␣100
120  SUM=SUM+A
130  IF␣A=10␣THEN␣EXIT␣FOR
140 NEXT␣A
```

制御変数Aが1から始まり100を超えるまで繰り返します。
制御変数Aの値を繰り返しごとに変数SUMに加えます。
制御変数Aの値が10になったら繰り返しを出ます。

・例2／STEPによる増分の設定を用いた例

```
100 SUM=0
110 FOR␣A=3␣TO␣1␣STEP␣-1
120  SUM=SUM+A
130  PRINT␣"A:";A,"SUM:";SUM
140 NEXT␣A
150 PRINT␣"SUM=";SUM
160 END
```

増分を-1に設定します。
制御変数Aの値を変数SUMに加えます。

＜実行結果＞

```
A:3  SUM:3
A:2  SUM:5
A:1  SUM:6
SUM=6
```

・例3／入れ子構造のFOR TO STEP～NEXT文を用いた九九表示プログラム

```
100 FOR␣A=1␣TO␣9
110   FOR␣B=1␣TO␣9
120     X=A*B
130     S$=STR$(X)
140     RES$=RES$+S$+" "
150   NEXT␣B
160   PRINT␣RES$
170   RES$=""
180 NEXT␣A
190 END
```

制御変数Aを用いて九九の段を制御します。
制御変数Bを用いて九九の列を制御します。
変数Xに九九の答えを代入します。
数値変数Xの値を文字列に変換し、文字列変数S$に代入します。
段ごとの九九を格納する文字列変数RES$に、答えが格納された文字列変数S$を追加します。
九九の次の数への繰り返しを継続させます。
終了した段の文字列を表示します。
格納された内容を空にして、次の段への繰り返しを継続します。
次の段へ繰り返しを継続します。

＜実行結果＞

```
1 2 3 4 5 6 7 8 9
2 4 6 8 10 12 14 16 18
3 6 9 12 15 18 21 24 27
4 8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```

■GOTO　指定行からの実行の継続

◆書式

GOTO *行番号*

◆解説

・*行番号*から、実行を続けます。
　指定する行番号は符号なしの整数で指定する必要があります。
・プログラムにない行番号を指定するとエラーになります。
・行番号にラベル名をつけることもできます。

◆関連項目

ON～GOTO (→p.19参照)　LABEL (→p.35参照)

・例／GOTOを用いた例

```
100 X=X+1
110 PRINT␣"*"
120 IF␣X<5␣THEN␣GOTO␣100
130 END
```

変数Xの値が5より小さい場合、行番号100に実行を移します。

＜実行結果＞

```
*
*
*
*
*
```

■IF THEN～END IF　条件による分岐

◆書式

(1)単純実行文の条件分岐
　IF *論理式* THEN { *命令文* | *行番号* } [ELSE { *命令文* | *行番号* }]

(2)複数実行文の条件分岐
　IF *論理式* THEN
　　[(一連の行)]
　[ELSE
　　(一連の行)]
　END IF

(3)ELSEIF区を持つ条件分岐
　IF *論理式1* THEN
　　[(一連の行)]
　ELSEIF *論理式2* THEN
　　[(一連の行)]
　[ELSE
　　(一連の行)]
　END IF

◆解説

・論理式の評価結果によって実行する文を選択します。
　論理式はひとつまたは複数を列挙することができます。
・*論理式*が真（論理式の条件に合う）の場合、THENに続く命令文を実行します。(→例1)
・*論理式*が偽（論理式の条件に合わない）の場合、ELSEがあればELSEに続く命令文を実行します。
　ELSEがない場合、IF文の次の行から実行を継続します。
・THEN句およびELSE句の直後に、複数の実行文を書くことはできません。複数の行を実行したい場合は、書式（2）を用います。
・IF文の *論理式1* が偽の場合、ELSEIF行へ実行が移ります。(→例3)
　ELSEIFに続く *論理式2* が真の場合、ELSEIF行のTHENに続く一連の行が実行されます。
　論理式2 が偽の場合、ELSEおよびそれに続く一連の行がある場合、それらが実行されます。
　ELSEおよびそれに続く一連の行がない場合、END IF行の次の行から実行が継続します。

◆関連項目

SELECT CASE (→p.23参照)

・例1

```
100 SCORE=50
110 IF SCORE>=70 THEN PRINT "合格" ELSE 130
120 END
130 PRINT "不合格"
140 END
```

＜実行結果＞
不合格

＜解説＞
110　変数SCOREが70と比べて等しいか大きい場合"**合格**"を表示し、そうでない場合 130 行に実行を移します。
130　110 行でELSE句が実行された場合、この行に実行が移り"**不合格**"を表示します。

・例2

```
100 FOOD=1
110 IF FOOD=1 THEN
120  CAL=500
130  PRINT "さばの味噌煮"
140 ELSE
150  CAL=650
160  PRINT "オムレツ"
170 END IF
180 END
```

変数FOODの値が数値1と等しければ120～130行のTHEN区を実行します。

変数FOODの値が数値1と等しくなければ150～160行のELSE区を実行します。

＜実行結果＞
さばの味噌煮

・例3

```
100 MATH=40
110 ENGLISH=60
120 IF MATH>=70 AND ENGLISH>=75 THEN
130  PRINT"合格"
140 ELSEIF MATH<50 OR ENGLISH<50 THEN
150  PRINT"追試"
160 ELSE
170  PRINT"及第"
180 END IF
190 END
```

＜実行結果＞
追試

＜解説＞
120　変数MATHの値が70以上、変数ENGLISHの値が75以上の場合、130行を実行します。
140　120行の論理式が偽でさらに、変数MATHの値が50未満、変数ENGLISHの値が50未満の場合、150行を実行します。
160　140行の論理式が偽であった場合、170行を実行します。

■INPUT　キーボードからの入力文字の取得

◆書式

INPUT ［PROMPT *文字列式*:］ *変数名* ［,*変数名2*,*変数名3*...］

◆解説

・キーボードから入力されたデータを変数に代入します。
・*変数名*が数値変数の場合、数値のみ代入することができます。
　文字列変数の場合は文字が代入できます。この場合、数字も文字列として扱われます。
・入力はENTERキーを押すことで終了します。
・PROMPTに続けて*文字列式*を書くことで、入力時に文字列を表示します。代入する*変数名*はコロン(:)に続けて書きます。
・コンマ(,)で変数名を区切ることで、複数の変数にキーボードの入力データを代入することができます。
　入力する際も同様にコンマ(,)で区切って入力します。
※この命令によるデータの入力をするにはUSB接続のキーボードがUSB端子に接続されている必要があります。

◆関連項目

INKEY$ (→p.101参照)　PAD (→p.104参照)

・例／キーボードからの入力の例

```
100 INPUT PROMPT"名前と点数を入力してください":NAME$,TEST1,test2,test3
110 PRINT NAME$;"さんの平均点:";(TEST1+TEST2+TEST3)/3
120 END
```

＜実行結果＞
名前と点数を入力してください
マリン,45,30,77 ↵
マリンさんの平均点:50.6667

＜解説＞
100　キーボードから入力を受け付け、変数NAME$,TEST1,TEST2,TEST3に代入します。
110　変数TEST1,TEST2,TEST3の平均を求めて表示します。

■LET　式の計算結果を変数に代入

◆書式

LET *変数名*［,*変数名2*,*変数名3*...］= *式*

◆解説

・*式*を計算し、その結果を*変数*に代入します。代入は次の順序にしたがって行われます。
　1.配列の添字(→p.6参照)や文字列変数の部分文字列指定(→p.5参照)を、変数名の並びの左から順番に評価
　2.右辺の式の評価
　3.式の結果を変数名の並びに対し左から代入
・LET命令はLETの語を省略して記述することができます。LETの語が省略してある場合、LET命令の行として扱われます。

・例1／LETを用いた例

```
LET P=3.14159
LET A,B,C=150
LET A(X,Y)=SIN(A)*N
LET A$="GREEN"
LET A$(2:4)="ABCDEF"
LET A$=B$ & "PLUS"
LET A$(2,3)(5:7)=$B
```

※上記はすべてLETの語を省略することができます。

数値変数Pに数値3.14159を代入します。
数値変数A,B,Cそれぞれに数値150を同時に代入します。
2次元数値配列Aに計算式の結果を代入します。
文字列変数A$に文字列"GREEN"を代入します。
文字列変数A$の2～4文字目までの部分文字列を、文字列"ABCDEF"で置き換えます。
文字列変数A$に、文字列変数B$の値と文字列"PLUS"をつないだ文字列を代入します。
2次元文字列配列A$の要素(2,3)の値の5～7文字目の値を、文字列変数B$の値で置き換えます。

・例2／LETの評価順序を示した例

```
100 LET␣A$="ABCDE"
110 LET␣A$(1:2),A$(2:4)= "XYZ"
120 PRINT␣A$
130 END
```

A$(1:2)="XYZ"を実行した後、A$(2:4)="XYZ"を実行します。

＜実行結果＞

```
XXYZDE
```

■ON～GOTO　指標を用いた行番号からの実行

◆書式

ON *指標* GOTO *行番号* [,*行番号*...]　[ELSE{ *命令文* | *行番号* }]

◆解説

・*指標*をもとに、GOTOに続く*行番号*の行から実行を継続します。
・GOTOに続く*行番号*の並びは、左端から1、2、3...と順番に番号が割り当てられます。その中から*指標*が示す番号と同じ順番の行番号が選び出され、その行から実行を継続します。
・ELSEに続けて*命令文*または*行番号*が書かれてあり、*指標*が0以下か*行番号*の並びの数より大きい場合は、ELSEに続く*命令文*の実行または*行番号*の行から実行を継続します。
・ELSE以降の文がない場合は、次の行から実行を継続します。

◆関連項目

GOTO (→p.15参照)　LABEL (→p.35参照)

・例／ON～GOTOを用いた例

```
100 A=2
110 ON␣A␣GOTO␣120,130,140␣ELSE␣150
120 PRINT␣"HOP"
130 PRINT␣"STEP"
140 PRINT␣"JUMP!"
150 END
```

変数Aの値2によって130行へ実行を移します。

＜実行結果＞

```
STEP
JUMP!
```

■OPTION BASE　下限値の設定

◆書式

OPTION BASE { *0* | *1* }

◆解説

・配列における暗黙の下限値を0または1に設定します。
・配列の添字は下限値を指定しない場合、暗黙的に1が設定されます。
・下限値の設定を実行した以降の、DIMによる配列宣言から有効となります。

◆関連項目

DIM (→p.11参照)

・例1

```
DIM A(10)
```

配列Aの添字の範囲は1～10。大きさは10。添字の下限値1は暗黙的に設定されます。

・例2

```
100 OPTION␣BASE␣0
110 DIM A(10)
120 END
```

下限値の値を0に設定します。
配列Aの添字の範囲は0～10。大きさは11になります。

■PRINT　テキストの表示

◆書式

PRINT 式 [{ , | ; }式...]

◆解説

・式を評価し、その値をテキストとして画面に表示します。
・式の間にコンマ(,)またはセミコロン(;)を用いることで、同じ行に式をまとめて書くことができます。
　コンマは、ふたつの空白に置き換えて表示します。セミコロンは表示の際、何も表示しません。
・数値を表示した場合、その数値に続けて1文字の空白が自動的に表示されます。

・例

```
100 PRINT␣128*64
110 X=80
120 PRINT␣X
130 PRINT␣12;34;"56"
140 PRINT
150 END
```

式128*64を評価し、その結果8192を表示します。

変数Xの値を表示します。
数値12、34 および文字列"56"を表示します。
何も表示せず、改行のみされます。

＜実行結果＞

```
8192
80
12  34  56
```
…何も表示されず改行のみされます。

■RANDOMIZE　乱数の初期化

◆書式

RANDOMIZE ［*乱数シード*］

項目	備考
乱数シード	範囲：-2e32～+2e32

◆解説

・RND命令(→p.98参照)で使用する乱数列を初期化します。
　RANDOMIZE命令を実行しないで乱数を発生させた場合、プログラムの実行のたびに常に同じパターンで値が生成されます。
・*乱数シード*を指定すると、それをもとに乱数列を生成します。
　*乱数シード*は数値を用います。
　指定した*乱数シード*から生成される乱数列は常に一定のパターンになります。
・*乱数シード*を指定せずに実行した場合、生成される乱数列は不定になります。

◆関連項目

RND (→p.98参照)

・例

```
100 PRINT␣RND
110 RANDOMIZE
120 PRINT␣RND
130 RANDOMIZE␣23
140 PRINT␣RND
150 END
```

RANDOMIZEを実行せず乱数を発生させると常に一定のパターンで値が発生します。
乱数シードを指定しないと不定の乱数列を生成します。

乱数シードに数値23を与えると、それをもとに乱数列を生成します。
乱数シードにより生成した乱数列から常に一定のパターンで値を発生させます。

＜実行結果＞

```
RUN↵
0.51387
0.774261
0.818885

RUN↵
0.51387
0.288125
0.818885
```

■READ/DATA　データの読み込み

◆書式

(1) データの読み込み
　READ *変数名* [,*変数名2,変数名3...*]
(2) データ並び
　DATA *データ* [,*データ2,データ3...*]

◆解説

・DATA文に書かれているデータ並びを、READ命令によって変数の並びに代入します。
　データには数値または文字列を用いることができます。
・READ文を実行すると、データがDATA文のデータ並びの先頭から順番に、READ文の変数名並びの*変数*へと代入されていきます。
・READ文の変数名の並びに対し、DATA文のデータの並びが足りない場合、エラーになります。
　変数名の並びが少ない場合は、余ったデータを無視します。
・DATA文は複数行にわたって書くことができます。DATA文が複数ある場合、データ並びはプログラム全体でひとつとして扱われます。
・DATA行では、プログラムは何もせず次の行へ実行を移します。

◆関連項目

RESTORE (→p.22参照)

・例

```
100 DATA␣1,2,3,4,5
110 READ␣A,B,C
120 READ␣D,E,F,G
130 DATA␣6,7,8
140 PRINT␣A,B,C,D,E,F,G
150 END
```

DATA 文自体は直接実行されず次の行に実行が移ります。
データ並びの先頭から3つのデータ1,2,3が、それぞれ変数A,B,Cに代入されます。
次の読み込み位置から4つのデータ4,5,6,7が、それぞれ変数D,E,F,Gに代入されます。
読み込まれなかったデータ8はそのまま無視されます。

＜実行結果＞

```
1   2   3   4   5   6   7
```

■RESTORE　データ読み込み位置の変更

◆書式

RESTORE ［*行番号*］

◆解説

・データの読み込み位置を、DATA命令で始まるデータ並びの先頭に戻します。
・*行番号*を指定した場合、指定した行の先頭のデータから読み込みます。

◆関連項目

READ (→p.21参照)　LABEL (→p.35参照)

・例

```
100 READ␣A,B
110 PRINT␣"1:",A,B
120 READ␣A,B,C
130 PRINT␣"2:",A,B,C
140 RESTORE
150 READ␣A,B,C,D
160 PRINT␣"3:",A,B,C,D
170 RESTORE␣200
180 READ␣A,B,C
190 PRINT␣"4:",A,B,C
200 DATA␣100,110,120
210 DATA␣130,140,150
220 END
```

200～210行のデータ並び先頭から、変数A,Bにデータを読み込みます。

現在のデータ読み込み位置から変数A,B,Cに順番にデータを読み込みます。

現在のデータ読み込み位置を先頭に戻します。
現在データ並び先頭にあるデータ読み込み位置からデータを読み込みます。

データ読み込み位置を200行の先頭にします。

＜実行結果＞

```
1:  100   110
2:  120   130   140
3:  100   110   120   130
4:  100   110   120
```

■REM　プログラムの注釈

◆書式

(1) REM *注釈文*
(2) ！*注釈文*

◆解説

・プログラムの注釈を書きます。
・プログラムの実行が注釈文に到達すると何もしないで次の行に進みます。
・REM行は何もせずに次の行へ実行を移します。(書式(1))
・注釈記号"!"から行末まで行末注釈として解釈され、何も実行せずに次の行へ実行を移します。(書式(2))

・例

```
100 REM␣キーコードテスト
110 DO␣!␣メインループ
120   PRINT␣KEYCODE1
130   !␣PRINT␣KEYCODE2
140 LOOP
150 END
```

注釈を用いてプログラムの機能を記述しています。
行末注釈を用いて、実行される行に注釈を加えています。

命令文を行末注釈化することで、実行を無視させます。

■SELECT CASE～/END SELECT　実行する行の選択

◆書式

SELECT CASE *式*
 CASE *照合項目* [,*照合項目...*]
　　(一連の行)
 [CASE *照合項目* [,*照合項目...*]
　　(一連の行)]
 ...
 [CASE ELSE
　　(一連の行)]
END SELECT

◆解説

・*式*を評価し、その結果とCASEに続く*照合項目*が一致した場合、CASE区内の一連の行が実行されます。
・*照合項目*に、以下の構文のいずれかを書くことができます。
(a)*定数*
(b)*定数* TO *定数*
(c)IS *比較演算子* *定数*
(a)～(c)以外の変数や式を書くとエラーになります。
・*照合項目*はコンマ(,)で区切って連続的に書くことができます。このとき、*照合項目*のいずれかに一致した場合、CASEに続く一連の行が実行されます。
・CASE区内の一連の行の実行を終了するか、または*照合項目*のどれにも一致せずCASE ELSE行がなかった場合、END SELECT行の次の行から実行を継続します。
・CASE ELSE行およびそれに続く一連の行を書くことによって、いずれの*照合項目*にも一致しない場合の実行を書くことができます。

◆関連項目

IF THEN～END IF (→p.16参照)

・例

```
100 INPUT␣A
110 SELECT␣CASE␣A
120 CASE␣77
130  PRINT␣"LUCKY!"
140 CASE␣40␣TO␣70,␣99
150  PRINT␣"CHANCE"
160 CASE␣IS␣<␣0
170  PRINT␣"TRY␣AGAIN"
180 CASE␣ELSE
190  PRINT␣"SPECIAL␣CHANCE"
200 END␣SELECT
210 END
```

変数Aの値によって実行する行を選択します。
値が77の場合

値が40～70の範囲または99の場合

値が0より小さい場合

上記CASE文のいずれにも当てはまらない場合

＜実行結果＞

```
?
-56↵
TRY AGAIN
```

※USB接続のキーボードが接続されている場合のみ、入力することができます。

■STOP　プログラムの中断

◆書式

STOP

◆解説

・プログラムの実行を中断します。
・CONTINUE命令(→p.25参照)によってプログラムを再開することができます。
・プログラム実行中、USBキーボードのBreakキーを押すか、STARTボタン+SELECTボタンを押しても中断することができます。

◆関連項目

CONTINUE (→p.25参照)

・例

```
100 DO
110  CODE=KEYCODE1                    押されているキーのコードを変数CODEに代入します
120  PRINT␣CODE
130  IF␣CODE=41␣THEN␣STOP            変数CODEの値が41(ESCキーが押された)の場合、STOP命令を実行します。
140 LOOP
150 END
```

＜実行結果＞

```
...
0
0
41
実行を中断しました。
```

直接命令

■CONTINUE　プログラムの再開

◆書式

CONTINUE

◆解説

・実行を中断したプログラムを再開します。
・STOP命令(→p.24参照)またはBreakキーなどで実行を中断した際、CONTINUE命令によって実行を再開することができます。
・現在中断しているプログラムを編集すると、CONTINUE命令を実行することができなくなります。

◆関連項目

STOP (→p.24参照)

■DELETE　プログラム行の削除

◆書式

DELETE *行範囲*

項目	備考
行範囲	書式:(a)*行番号* (b)*開始行* TO *終了行*
開始行	書式:{ *行番号* | FIRST | LAST }
終了行	書式:{ *行番号* | FIRST | LAST }

◆解説

・*行範囲*で指定した範囲の行を削除します。
・*行範囲*には以下の指定方法があります。
(a)行番号
　*行番号*を指定すると、その行を削除します。
(b)開始行 TO 終了行
　*開始行*から*終了行*までの行を削除します。
　FIRSTはプログラムの先頭行を意味します。
　LASTはプログラムの最終行を意味します。
・行範囲に*開始行*と*終了行*を指定する場合、*終了行*は*開始行*と同じかそれ以降の行を指定しなければなりません。

・例

```
DELETE 120
DELETE 240␣TO␣320
DELETE FIRST␣TO␣500
DELETE 200␣TO␣LAST
DELETE FIRST␣TO␣LAST
```

120行を削除します。
240行～320行までを削除します。
プログラム先頭行から500行までを削除します。
200行からプログラム最終行までを削除します。
プログラムをすべて削除します。

■FILES　ディレクトリ内のファイルの一覧表示

◆書式

FILES ［*ディレクトリ名*］

◆解説

・データが保存されているディレクトリ内のファイルを一覧表示します。
・指定したディレクトリ内にあるファイルの一覧を表示します。
・*ディレクトリ名*を省略すると、ディレクトリの一覧を表示します。

◆関連項目

KILL (→p.26参照)　SAVE (→p.30参照)
LOAD (→p.28参照)　DEVICE (→p.33参照)

・例

```
FILES                          ディレクトリの一覧を表示します。
FILES "USER"                   "USER"ディレクトリ内のファイルの一覧を表示します。
```

■FKEY　ファンクションキーへの文字列の登録

◆書式

FKEY *文字列*, *ファンクションキー番号*

項目	備考
文字列	範囲:0～30文字
ファンクションキー番号	範囲:1～12

◆解説

・ファンクションキーを押したときに表示する文字列を設定します。
・*ファンクションキー番号*に対応した番号を持つファンクションキーを押すと、*文字列*を表示します。

・例

```
FKEY "RENUMBER␣AT␣1000",6     F6キーを押すことで"RENUMBER AT 1000"をテキスト画面に表示します。
```

■KILL　ファイルの削除

◆書式

KILL ［*ディレクトリ名*/］*ファイル名*

◆解説

・*ファイル名*で指定したファイルを削除します。
・ディレクトリ内のファイルを削除したい場合は、ディレクトリ名とスラッシュ(/)に続けてファイル名を書きます。
　*ディレクトリ名*を省略すると"SYSTEM"ディレクトリが対象となります。

◆関連項目

FILES (→p.26参照)　LOAD (→p.28参照)　SAVE (→p.30参照)

■LIST　プログラム内容の表示

◆書式

LIST ［*行範囲*］ ［REF キーワード］

項目	備考
行範囲	書式:(a)*行番号* (b)*開始行* TO *終了行*
開始行	書式:{ *行番号* | FIRST | LAST }
終了行	書式:{ *行番号* | FIRST | LAST }

◆解説

・プログラムを画面に表示します。
・*行範囲*を指定することでプログラムの表示範囲を指定することができます。
　指定しない場合、プログラムをすべて表示します。
・*行範囲*は以下の指定方法があります。
(a)行番号
　*行番号*を指定すると、その行を表示します。
(b)*開始行* TO *終了行*
　*開始行*から*終了行*までの行を表示します。
　FIRSTはプログラムの先頭行を意味します。
　LASTはプログラムの最終行を意味します。
・REFに続けてキーワードを書くと、キーワードが含まれる行を検索して表示します。
　キーワードは大文字と小文字を区別します。
　*行範囲*を指定した場合、その行範囲でキーワードが含まれる行を表示します。

◆関連項目

VIEW (→p.31参照)

・例

```
LIST
LIST FIRST␣TO␣400
LIST REF␣"REM"
LIST 200␣TO␣400␣REF␣"<>"
```

プログラムをすべて表示します。
プログラムの先頭行から400行までを表示します。
プログラムのすべての行で文字列"REM"を含む行を表示します。
200行～400行の範囲で文字列"<>"を含む行を表示します。

■LOAD　プログラムの読み込み

◆書式

LOAD ［*ディレクトリ名/*］*ファイル名* ［MERGE］

◆解説

・指定したデバイスから、*ファイル名*に指定したプログラムファイルを読み込みます。
・現在編集しているプログラムは破棄されます。
・DEVICE命令(→p.33参照)により、ファイルを読み込むデバイスを設定することができます。
・*ディレクトリ名*を指定すると、そのディレクトリ内から*ファイル名*に指定したプログラムファイルを読み込みます。
　*ディレクトリ名*を省略すると、"SYSTEM"ディレクトリ内からファイルを読み込みます。
・MERGEを指定すると、現在編集しているプログラムと読み込むプログラムをつなげます。行番号が同じ行は、読み込むプログラムに上書きされます。

◆関連項目

DEVICE (→p.33参照)　SAVE (→p.30参照)
FILES (→p.26参照)　KILL (→p.26参照)

・例

```
LOAD "TEST"
LOAD "USER/GAME1"
```

指定したデバイスの"SYSTEM" ディレクトリから、プログラムファイル"TEST"を読み込みます。
指定したデバイスの"USER"ディレクトリから、プログラムファイル "GAME1"を読み込みます。

■MEMORY　空きメインメモリ量の表示

◆書式

MEMORY

◆解説

・メインメモリの残り量を表示します。

■NEW　新規プログラムの作成

◆書式

NEW

◆解説

・現在のプログラムを破棄して、新たにプログラムを編集します。
・使用されていた変数、配列も同時に破棄されます。

◆関連項目

INIT (→p.34参照)

■QUIT　"EZ-BASIC"の終了

◆書式

QUIT

◆解説

・"EZ-BASIC"を終了します。
・現在編集中のプログラムは破棄されます。

■RENUMBER　行番号への再番号付け

◆書式

RENUMBER［*行範囲*］［AT *新規開始行*］［STEP *刻み*］

項目	備考
行範囲	書式:(a)*行番号* (b)*開始行* TO *終了行*
開始行	書式:{ *行番号* | FIRST | LAST }
終了行	書式:{ *行番号* | FIRST | LAST }

◆解説

・プログラムの行番号に対し、新しい行番号を割り当てます。
・*行範囲*を指定することでプログラムの行番号の割り当て範囲を指定します。
　*行範囲*を指定しない場合、プログラム全体をRENUMBER命令の対象とします。
・*行範囲*は以下の指定方法があります。
(a)行番号
　*行番号*を指定すると、その行を再番号付けします。
(b)開始行 TO 終了行
　*開始行*から*終了行*までの行を再番号付けします。
　FIRSTはプログラムの先頭行を意味します。
　LASTはプログラムの最終行を意味します。
・ATに続けて*新規開始行*を指定することで、新しく割り当てる最初の行番号を指定します。
　*新規開始行*を指定しない場合、100を割り当てます。
・STEPに続けて*刻み*を指定することで、新しく割り当てる行番号の増加量を指定します。
　*刻み*を指定しない場合、10を割り当てます。
・新しく割り当てられる行番号が、既にある行番号と重複あるいはまたぐ場合、実行することができません。

◆関連項目

LIST (→p.27参照)　　VIEW (→p.31参照)

・例

```
RENUMBER                    プログラム内のすべての行番号を、行番号100から始め10刻みで新しく割り当てます。
RENUMBER 300␣TO␣LAST        プログラムの行番号300から最終行までの範囲を、行番号100から始め10刻みで新しく行番号を割り当てます。
RENUMBER AT␣1000␣STEP␣25    プログラム内のすべての行番号を、行番号1000から始め25刻みで新しく割り当てます。
```

■RUN　プログラムの実行

◆書式

RUN

◆解説

・プログラムを実行します。
・プログラムの実行をSTOP命令(→p.24参照)またはUSBキーボードのBreakキー、STARTボタン+SELECTボタンで中断することができます。

◆関連項目

STOP (→p.24参照)　CONTINUE (→p.25参照)　END (→p.13参照)

■SAVE　プログラムの保存

◆書式

SAVE ［*ディレクトリ名*/］*ファイル名*

◆解説

・現在編集しているプログラムを保存します。
・*ファイル名*にプログラムを保存するファイルの名前を指定します。
　*ファイル名*は8文字までの英数字および下線(_)のみ使用することができます。
・ディレクトリを指定してファイルを保存したい場合は、ディレクトリ名とスラッシュ(/)に続けてファイル名を書きます。
　ディレクトリがない場合、新しく作成します。
　*ディレクトリ名*は最大8文字までつけることができます。
・*ディレクトリ名*を省略すると、"SYSTEM"ディレクトリに保存します。
・ひとつのディレクトリに最大15個のファイルを作成することができます。

◆関連項目

FILES (→p.26参照)　KILL (→p.26参照)　LOAD (→p.28参照)

・例

```
SAVE "FILE01"
SAVE "USER/SAMPLE"
```

現在のプログラムを、"SYSTEM"ディレクトリ内のファイル"FILE01"に保存します。
現在のプログラムを、ディレクトリ"USER"内にファイル名"SAMPLE"として保存します。
ディレクトリ"USER"がない場合、ディレクトリが新規に作成されます。

■VIEW　プログラム内容の表示

◆書式

VIEW ［*行範囲*］

項目	備考
行範囲	書式:(a)*行番号* (b)*開始行* TO *終了行*
開始行	書式:{ *行番号* | FIRST | LAST }
終了行	書式:{ *行番号* | FIRST | LAST }

◆解説

・プログラムを画面に表示します。
・*行範囲*を指定することでプログラムの表示範囲を指定することができます。
・*行範囲*は以下の指定方法があります。
(a)行番号
　*行番号*を指定すると、その行を画面に表示します。
(b)開始行 TO 終了行
　*開始行*から*終了行*までの行を画面に表示します。
　FIRSTはプログラムの先頭行を意味します。
　LASTはプログラムの最終行を意味します。
・*行範囲*に*開始行*と*終了行*を指定する場合、*終了行*は*開始行*と同じかそれ以降の行を指定しなければなりません。
・プログラムの表示が画面におさまらない場合、カーソルキーの上下およびPage Upキー、Page Downキーでプログラムの表示をスクロールさせることができます。
・BreakキーまたはEnterキーを押すとVIEW命令による表示を終了します。

◆関連項目

LIST (→p.27参照)

・例

```
VIEW                    プログラムをすべて表示します。
VIEW 200␣TO␣500         200行～500行までのプログラムを表示します。
VIEW FIRST␣TO␣400       プログラムの最初から400まで表示します。
```

■グラフィックス仕様

◆スクリーン構成

・スクリーン(画面)はテキスト画面およびグラフィックス画面から構成されます。

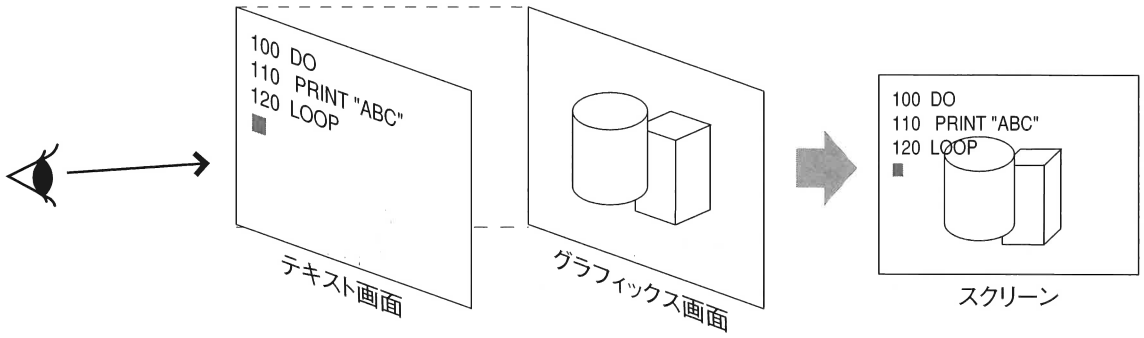


図4：スクリーン構成

◆テキスト画面

・テキストを表示するための画面です。モデルやプリミティブより常に手前に表示されます。

◆グラフィックス画面

・表示エリアは横640ドット、縦224ラインです。
・Zバッファ方式(→p.44参照）により、各ピクセルはZ値を持ちます。
・RGB各256階調、16,777,216色が表示可能です。
・各色は、赤成分、緑成分、青成分の合成によって得られる色によって表現されます。各成分は0～255の値によって指定します。0を指定すると色成分が最も弱く、255を指定すると色成分が最も強くなります。
・アルファ値によって透明度を指定することができます。
アルファ値は0～128の値によって指定します。128を指定すると完全に不透明に、0を指定すると完全に透明になります。
・テレビ画面に表示されるピクセルは、横方向に比べ縦方向が2倍の長さを持ちます。

◆色指定

・テキストカラーおよびマウスカーソルの色は、この色指定によって指定します。

■色指定

色指定	色	色指定	色
RED	赤	BLUE	青
CYAN	水色	PURPLE	紫
YELLOW	黄	GREEN	緑
BLACK	黒	WHITE	白

・テキストカラーはTEXTCOL命令(→p.40参照)、マウスカーソルの色はMOUCOL命令(→p.36参照)によって設定します。

■CLEAR　テキスト画面のクリア

◆書式

CLEAR

◆解説

・テキスト画面上に表示されている文字をすべて消去します。
・プリミティブおよびモデルには影響ありません。

■DEVICE　読み書き用デバイスの設定

◆書式

DEVICE *デバイス名*

◆解説

・データの読み込みおよび書き込みを行う対象となるデバイスを指定します。
・*デバイス名*に、読み書きの対象とするデバイス名を書きます。デバイス名は以下の通りです。

■デバイス一覧

デバイス名	デバイス
MEMC1	MEMORY CARD差込口1に接続された"PlayStation 2"専用メモリーカード(8MB)
MEMC2	MEMORY CARD差込口2に接続された"PlayStation 2"専用メモリーカード(8MB)

・初期状態ではMEMC1が設定されています。

◆関連項目

LOAD (→p.28参照)　SAVE (→p.30参照)　FILES (→p.26参照)
VARLOAD (→p.41参照)　VARSAVE (→p.41参照)　MODLOAD (→p.78参照)
TEXLOAD (→p.55参照)　PALLOAD (→p.50参照)
BGMLOAD (→p.82参照)　SELOAD (→p.83参照)

・例

```
DEVICE MEMC2
```

データを読み書きする対象を、MEMORY CARD差込口2側の"PlayStation 2"専用メモリーカード(8MB)に設定します。

■FSYNC　テレビ画面の表示を待つ

◆書式

FSYNC *待ち時間*

項目	備考
待ち時間	範囲：1以上

◆解説

・テレビ画面の表示が終了するまで、プログラムを一時停止します。
・プログラムを停止する時間を、*待ち時間*で指定します。
　*待ち時間*は、前回FSYNC命令を実行してから1回画面を描き終えるまでを単位とします。
・通常、*待ち時間*を1にすることで、一般的なテレビ画面の表示とプログラムの実行との同期を取ることができます。
・プログラムの一時停止が終了すると、次の行から実行を継続します。

◆関連項目

STOP (→p.24参照)

・例／FSYNCの動作を見る例

```
100 DO
110  CLEAR
120  PRINT␣"おかし！"
130  IF␣PAD(1,6)>0␣THEN␣FSYNC␣1
140 LOOP
```

＜解説＞
130　○ボタンが押されている間はFSYNC命令を実行します。
　　FSYNC命令を実行しない場合、画面に表示される文字がちらつきますが、○ボタンを押し続けてFSYNC命令を実行すると、
　　テレビの描画と同期を取るため画面に表示される文字がちらつきません。

■INIT　EZ-BASICの環境の初期化

◆書式

INIT [*ALL*]

◆解説

・EZ-BASICの環境を初期化します。
・初期化される項目は、プリミティブ、モデル、フォグ、カメラ、エフェクト、テキスト画面、サウンドの各種設定です。
・*ALL*を書くと、"EZ-BASIC"の環境すべてを初期化し、CD-ROMから各種バッファに初期設定のデータを読み込みます。
・編集中のプログラムは、初期化されずに残ります。

■LABEL　ラベルによる行の指定

◆書式

(1) LABEL *ラベル名*

(2) **ラベル名*

◆解説

・行にラベル名をつけます。
・GOTO命令やRESTORE命令などで行番号指定をする際、行番号の代わりに*ラベル名*を扱うことができます。
　*ラベル名*の先頭にアスタリスク(*)をつけることで、*ラベル名*を用いて行の指定をします。
・LISTやRENUMBERなどの直接命令系の行指定には使用することができません。

◆関連項目

GOTO (→p.15参照)　　ON～GOTO (→p.19参照)
RESTORE (→p.22参照)　　IF THEN～END IF (→p.16参照)

・例／ラベルを用いた例

```
100 A=2
110 ON␣A␣GOTO␣*HOP,*STEP,*JUMP␣ELSE␣*EXIT
120 LABEL␣HOP
130 PRINT␣"HOP"
140 LABEL␣STEP
150 PRINT␣"STEP"
160 LABEL␣JUMP
170 PRINT␣"JUMP!"
180 LABEL␣EXIT
190 PRINT␣"END"
200 END
```

変数Aの値2により2番目のラベルSTEPへ実行を移します。

110行の命令によりこの行に実行が移ります。

＜実行結果＞

```
STEP
JUMP!
END
```

■LOCATE　テキストカーソルの移動

◆書式

LOCATE *TextX*, *TextY*

項目	内容	備考
TextX	テキストX座標値	範囲：0～37(テキストモード1)、0～49(テキストモード2)
TextY	テキストY座標値	範囲：0～22

◆解説

・テキストカーソルを、テキスト座標(*TextX*, *TextY*)で指定した位置に移動します。
・座標の取り得る範囲以外の値を指定するとエラーになります。

・例

```
LOCATE 30,20
```

テキストカーソルをテキスト座標(30,20)に移動します。

■MOUAREA　マウスカーソルの移動範囲の設定

◆書式

MOUAREA *StartX*,*StartY*,*EndX*,*EndY*

項目	内容	備考
StartX	左上X座標	範囲：0～639
StartY	左上Y座標	範囲：0～223
EndX	右下X座標	範囲：0～639
EndY	右下Y座標	範囲：0～223

◆解説

・マウスカーソルの移動可能な範囲を、(*StartX*,*StartY*)～(*EndX*,*EndY*)で指定した矩形領域内のみに限定します。
・*StartX*>*EndX*または*StartY*>*EndY*の場合エラーになります。

◆関連項目

MOUDISP (→p.36参照)　MOUOFF (→p.37参照)　MOUGET (→p.37参照)

・例

```
MOUAREA 320,112,639,223
```

マウスカーソルの移動可能な範囲をX座標値320～639、Y座標値112～223に設定します。

■MOUCOL　マウスカーソルの色の設定

◆書式

MOUCOL *色指定*

項目	備考
色指定	(→p.32参照)

◆解説

・マウスカーソルの色を設定します。

◆関連項目

MOUAREA (→p.36参照)　MOUDISP (→p.36参照)
MOUOFF (→p.37参照)　MOUGET (→p.37参照)

・例

```
MOUCOL RED
```

マウスカーソルを赤に設定します。

■MOUDISP　マウスカーソルの表示

◆書式

MOUDISP

◆解説

・マウスカーソルを表示状態にします。

◆関連項目

MOUAREA (→p.36参照)　MOUOFF (→p.37参照)　MOUGET (→p.37参照)

■MOUGET　マウスカーソルの位置およびボタンの状態の取得

◆書式

MOUGET *PosX*,*PosY*,*Button1*,*Button2*

項目	内容	備考
PosX	X座標取得変数名	
PosY	Y座標取得変数名	
Button1	左ボタン状態取得変数名	
Button2	右ボタン状態取得変数名	

◆解説

・マウス左ボタンが押されている場合*Button1*の値が1に、マウス右ボタンが押されている場合*Button2*の値が1になります。ボタンが押されていない場合、値が0になります。
・現在マウスカーソルが表示されているスクリーン上の座標値を取得変数*PosX*,*PosY*に代入します。

◆関連項目

MOUAREA (→p.36参照)　MOUDISP (→p.36参照)　MOUOFF (→p.37参照)

・例

```
100 DO
110  MOUGET␣MX,MY,M1,M2
120  PRINT␣MX,MY,M1,M2
130 LOOP
```

マウスカーソルの位置を変数MX,MYに、現在のマウスボタンの状態を変数M1,M2に代入します。

■MOUOFF　マウスカーソルの非表示

◆書式

MOUOFF

◆解説

・マウスカーソルを非表示にします。
・カーソルは非表示になりますが、マウスの状態は更新されます。

◆関連項目

MOUAREA (→p.36参照)　MOUDISP (→p.36参照)　MOUGET (→p.37参照)

■MOUPAT　マウスカーソルの変更

◆書式

MOUPAT *Pat*

項目	内容	備考
Pat	カーソルパターン	範囲：1～18

◆解説

・マウスカーソルの形状を*Pat*に指定したパターンに変更します。

■MOUSET　マウスカーソルの移動

◆書式

MOUSET *MouseX*,*MouseY*

項目	内容	備考
MouseX	カーソルX座標値	範囲：0～639
MouseY	カーソルY座標値	範囲：0～223

◆解説

・マウスカーソルをスクリーン座標値（*MouseX*,*MouseY*）に移動します。

◆関連項目

MOUAREA (→p.36参照)　MOUDISP (→p.36参照)
MOUGET (→p.37参照)　MOUOFF (→p.37参照)

・例

```
MOUSET 320,112
```

マウスカーソルを画面中央に移動します。

■ORDER　描画順序の設定

◆書式

ORDER *描画モード*

項目	備考
描画モード	範囲：1～2

◆解説

・モデルとプリミティブの描画順序を設定します。
・*描画モード*に1を設定すると、先にモデルを描画し次にプリミティブを描画します。
　*描画モード*に2を設定すると、先にプリミティブを描画し次にモデルを描画します。

◆関連項目

ZSORT (→p.81参照)

・例／描画モードを切り替える例

```
100 MODDISP␣1
110 TILEDISP␣1
120 TILECOL␣1,255,0,0,64
130 TILEPOS␣1,200,50,440,224-50,800000
140 CAMPOS␣1,0,0,-50
150 DO
160  CLEAR
170  PRINT␣"PRESS START button"
180  D=PAD(1,20)+1
190  ORDER␣D
200  FSYNC␣1
210 LOOP
```

プリミティブをZ値800000の位置に置きます。

STARTボタンを押すと描画モードが2になります。
STARTボタンを離すと描画モードが1になります。

■SCREEN　画面モードの設定

◆書式

SCREEN *画面モード*

◆解説

・画面の分割方法を設定します。
・画面モードにより、以下の画面分割ができます。

■画面モードの種類

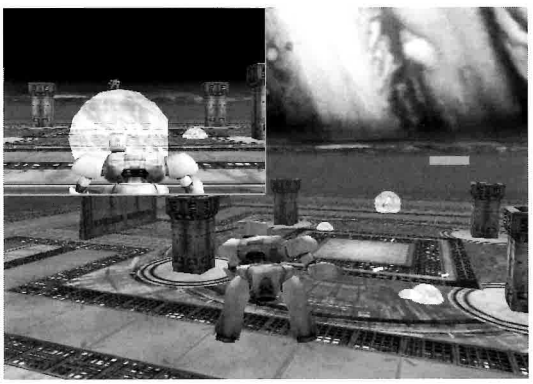
画面モード	効果
SINGLE	画面を1画面表示にします。1番のカメラによる映像が表示されます。
SPLIT	画面を左右に2分割します。左がカメラ番号1、右がカメラ番号2のカメラによる映像になります。
VERTICAL	画面を上下に2分割します。上がカメラ番号1、下がカメラ番号2のカメラによる映像になります。
WINDOW StartX, StartY,EndX,EndY	(*StartX*, *StartY*)～(*EndX*, *EndY*)で指定したスクリーン上の矩形領域にカメラ番号2のカメラで撮影した映像が表示されます。

・画面分割は、モデルの表示に影響します。プリミティブおよびテキスト表示には影響しません。

・例

```
SCREEN WINDOW 0,0,300,120
```

カメラ1が撮影した画面上の、スクリーン座標上の(0,0)を左上(300,120)を右下とする矩形領域に、カメラ番号2のカメラで撮影した映像を表示します。



■TEXTCOL　テキストの色の設定

◆書式

TEXTCOL *色指定*

項目	備考
色指定	(→p.32参照)

◆解説

・テキストを表示する色を設定します。

◆関連項目

PRINT (→p.20参照)

・例

```
TEXTCOL BLUE
```

テキストが青で表示されるようになります。

■TEXTMODE　テキストモードの設定

◆書式

TEXTMODE *モード値*

項目	備考
モード値	範囲：1～2

◆解説

・テキスト画面の表示文字数を設定します。
・*モード値*により、テキスト画面の表示可能文字数を設定できます。

■テキストモード

モード値	表示文字数
1	横38文字、縦23文字
2	横50文字、縦23文字

・テキストモードを設定すると、それまで表示されていたテキスト画面がクリアされます。

・例

```
TEXTMODE 2
```

テキストモードを横50文字、縦23文字のモードにします。

■VARLOAD　ファイルから変数へのデータ読み込み

◆書式

VARLOAD ［*ディレクトリ名/*］*ファイル名,変数名*

◆解説

・ファイルからデータを読み込み、変数に代入します。
・ファイルには、拡張子がVRNの数値変数ファイルおよび拡張子がVRSの文字列変数ファイルを指定することができます。
・変数には配列を指定することもできます。
・読み込むデータの型と変数の型が一致していないとエラーになります。
・*ディレクトリ名*を指定すると、そのディレクトリ内からファイルを読み込みます。
　*ディレクトリ名*を省略すると、"SYSTEM"ディレクトリからファイルを読み込みます。

◆解説

VARSAVE (→p.41参照)

・例

```
VARLOAD "SAVE01",DATA01
```

"SYSTEM"ディレクトリ内にあるファイル"SAVE01"内のデータを、変数DATA01に読み込みます。

■VARSAVE　変数内容のファイルへの書き込み

◆書式

VARSAVE ［*ディレクトリ名/*］*ファイル名,変数名*

◆解説

・変数の値をファイルに書き込みます。
・変数には配列を指定することもできます。配列の場合、配列変数名を指定します。
　このとき、配列の要素の値すべてをファイルに保存します。
　数値変数を指定すると拡張子がVRNのファイルに書き込み、文字列変数を指定すると拡張子がVRSのファイルに書き込みます。
・ディレクトリ内にファイルを保存したい場合は、*ディレクトリ名*とスラッシュ（/）に続けて*ファイル名*を書きます。
　指定したディレクトリがない場合、新しく作成します。
・*ディレクトリ名*および*ファイル名*には、最大8文字までの英数字および下線（_）が使用できます。
　*ディレクトリ名*を省略すると、"SYSTEM"ディレクトリ内にファイルを保存します。
・ひとつのディレクトリに最大15個のファイルを作成することができます。

◆関連項目

VARLOAD (→p.41参照)

・例1

```
VARSAVE "SAVE/DATA1",A
VARSAVE "DATA2",B
```

"SAVE"ディレクトリ内のファイル"DATA1"に、変数Aの値を保存します。
変数Bは配列とします。この場合、"SYSTEM"ディレクトリ内のファイル"DATA2"に
配列の要素すべてを保存します。

・例2

```
100 SAV01=234
110 PRINT␣"セーブしてよろしいですか?"
120 DO
130   IF␣PAD(1,6)>0 THEN
140     VARSAVE␣"SAVE01",SAV01
150     PRINT␣"セーブしました。"
160     EXIT␣DO
170   ELSEIF␣PAD(1,7)>0␣THEN
180     PRINT␣"セーブ中断しました。"
190     EXIT␣DO
200   END␣IF
210 LOOP
220 END
```

○ボタンが押されたら
変数SAV01の値をファイル"SAVE01"に保存

×ボタンが押されたら中断

■VIBRATE　コントローラの振動

◆書式

VIBRATE *ContNo,VibrateL,VibrateR,Time*

項目	内容	備考
ContNo	コントローラ番号	範囲：1～2
VibrateL	左振動モーター	範囲：0～255
VibrateR	右振動モーター	範囲：0～1
Time	振動時間(秒)	範囲：0～10

◆解説

・アナログコントローラ(DUALSHOCK 2)を振動させます。
・ContNoに、振動させるコントローラが接続されたコントローラ端子の番号を指定します。
・アナログコントローラ(DUALSHOCK 2)には内部に特性の異なる左右ふたつのモーターが内蔵されており、その振動を制御することができます。
　VibrateLの値により左側振動モーターの振動の強さを指定します。
　VibrateRの値により右側振動モーターの停止または振動を指定します。
・Timeに振動させる時間を指定します。
　振動を停止するにはTimeに0を指定します。
・各値に範囲以外の値を指定するとエラーになります。
※コントローラの寿命を早める可能性があるため、長時間の振動はさせないでください。

・例

```
VIBRATE 1,255,1,5
```

コントローラ端子1に接続されたアナログコントローラ(DUALSHOCK 2)を、5秒間強く振動させます。

2Dグラフィックス関連

■スクリーンについて

◆スクリーン座標系

テレビに表示される、スクリーンの座標です。

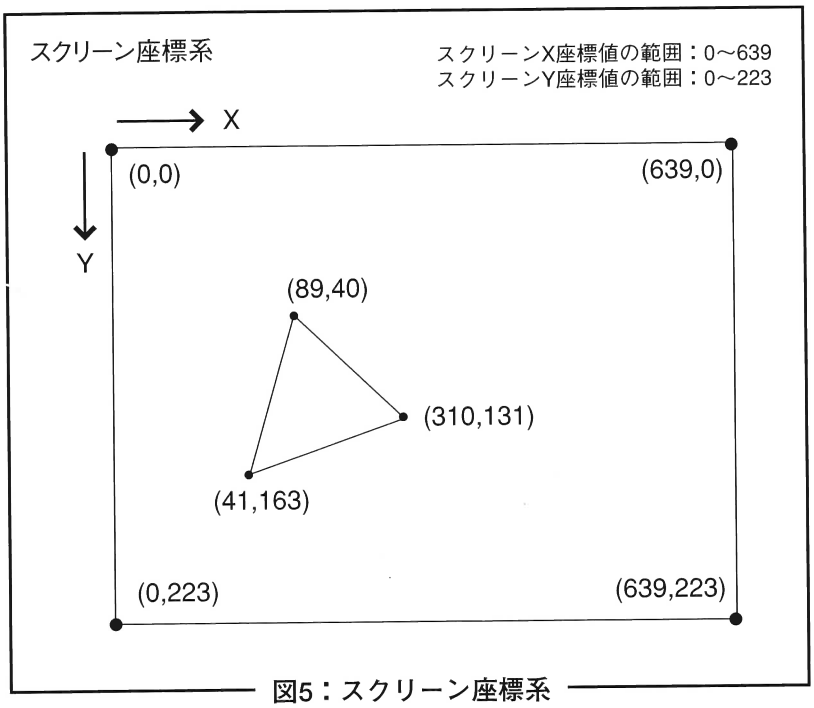


図5：スクリーン座標系

◆プリミティブ

・描画時の最小単位として扱われるグラフィックスです。プリミティブにはライン、タイル、トライアングル、スプライトの4種類があります。
・プリミティブの描画における前後関係は、Z値によって設定します。
Z値は0～16777200までの値で指定します。0が描画時最も奥に配置され、16777200が描画時最も手前に配置されます。
・各プリミティブは通常、プリミティブ番号順に描画されます。
・3Dグラフィックスのカメラ、ライト、フォグの影響はありません。

◆テキスト

・画面上に表示できる文字は、テキストモード(→p.40参照)が1の場合、横38文字縦23文字です。テキストモードが2の場合、横50文字縦23文字です。
・全角かな漢字は、1文字につき横方向へ半角文字2文字分の表示幅が必要です。
・Z値は設定できません。常に画面手前に表示されます。
・3Dグラフィックスのカメラ、ライト、フォグの影響はありません。

◆マウスカーソル

・USBマウス接続時にはマウスによって操作することができます。また、プログラム上で表示・非表示を切り替えたりカーソルの移動範囲を限定することができます。
・3Dグラフィックスのカメラ、ライト、フォグの影響はありません。

■Zバッファ方式

・スクリーンを構成する各ピクセルが奥行き値(Z値)を持つ方式です。
Z値を持つことで、ふたつのプリミティブの前後関係を調べ、適切な描画を行うことができます。

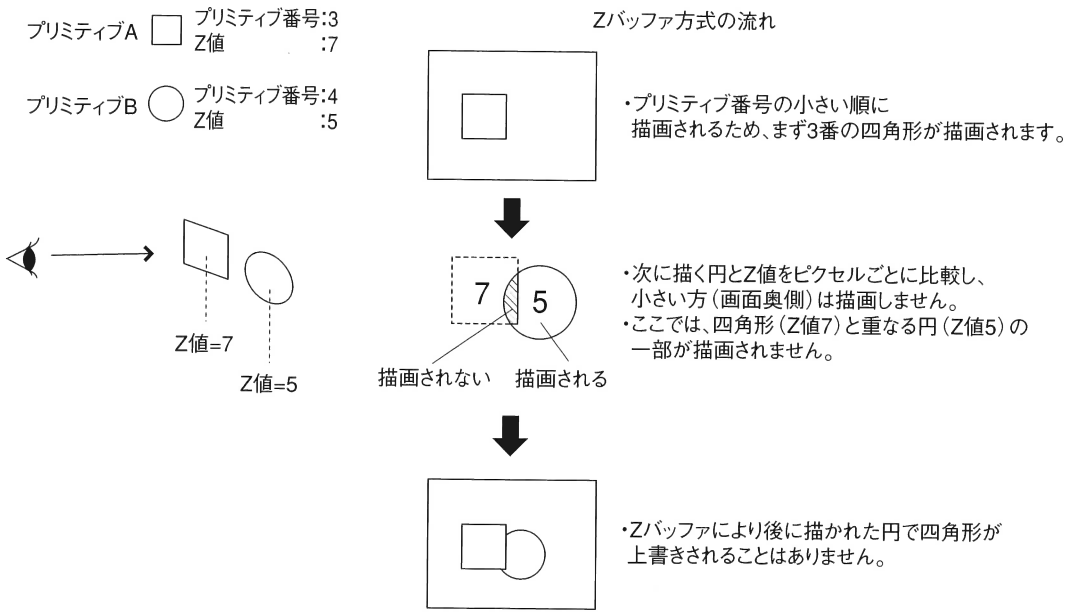


図6：Zバッファ方式

■Zバッファ問題

・半透明のモデルを使用している場合、Zバッファ問題が発生することがあります。
これは、手前の半透明モデルと奥のオブジェクトが重なっている部分が、Zバッファの機能により描画されないため、重なっているはずの半透明部分に何も描画されなくなる現象を指します。この問題を軽減する手段として、Zソート(→p.45参照)を用います。

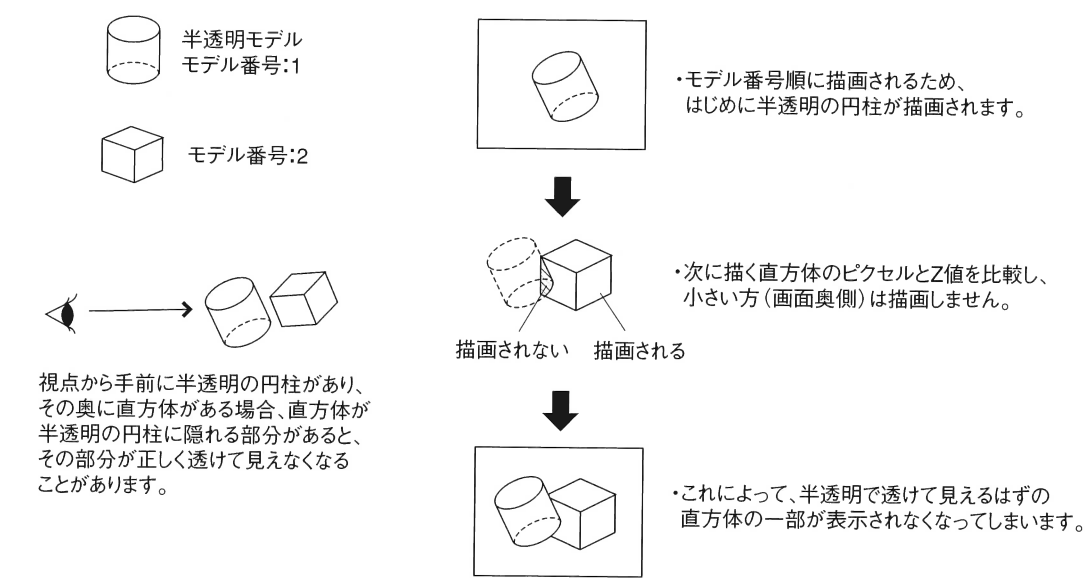


図7：Zバッファ問題

■Zソート

・Z値によりモデルをソート(並び替え)することで、あらかじめ前後関係を決定しておくことができます。これにより重なる半透明部分に適切な描画が行われます。
・ZソートによりZバッファ問題をほぼ回避することができますが、一方でZソートによる計算負荷が増加します。
・Zソートはモデルの原点を単位に行われるため、モデルの形によっては適切に描画されないなど、Zバッファ問題が完全に解決できない場合があります。
・Zソートを行う場合は、ZSORT命令 (→p.81参照)でソートするモデルを設定しておきます。

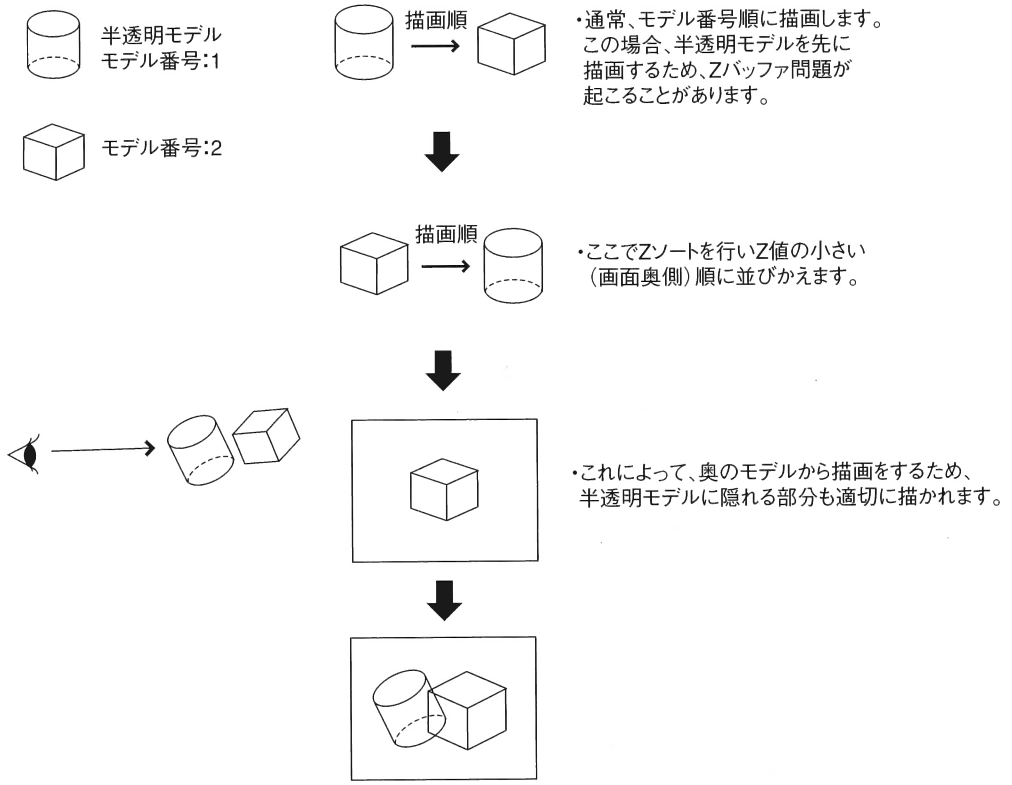


図8：Zソート

■BGCOL　背景色の設定

◆書式

BGCOL *Red,Green,Blue*

項目	内容	備考
Red	赤成分	範囲：0～255
Green	緑成分	範囲：0～255
Blue	青成分	範囲：0～255

◆解説

・画面の背景色を設定します。
・*Red,Green,Blue*の各値に色成分を設定します。
・各値に範囲以外の値を指定するとエラーになります。

・例

```
BGCOL 255,0,0          背景色が赤になります。
BGCOL 0,255,255        背景色が水色になります。
BGCOL 200,133,43       背景色が橙色になります。
```

■BSCOL　ScrollLock時のスクリーン色の設定

◆書式

BSCOL *Red,Green,Blue,Alpha*

項目	内容	備考
Red	赤成分	範囲：0～255
Green	緑成分	範囲：0～255
Blue	青成分	範囲：0～255
Alpha	アルファ値	範囲：0～128

◆解説

・ScrollLock有効時のスクリーン色を設定します。
・ScrollLockキーを押してScrollLockを有効にすることができます。
　この状態でもう一度ScrollLockキーを押すことで、ScrollLockを解除することができます。
・*Red,Green,Blue*で色を指定し、*Alpha*で透明度を指定します。

・例

```
BSCOL 128,0,0,96       ScrollLock キーを押すとスクリーン色を赤い半透明にします。
```

■EFFECT　残像効果

◆書式

EFFECT *Alpha,StartX,StartY,EndX,EndY,Z*

項目	内容	備考
Alpha	アルファ値	範囲：0～127
StartX	始点X座標値	範囲：-1728～2367
StartY	始点Y座標値	範囲：-1936～2159
EndX	終点X座標値	範囲：-1728～2367
EndY	終点Y座標値	範囲：-1936～2159
Z	奥行き値	範囲：0～16777200

◆解説

・描画した画像を、次の描画まで残して表示する設定をします。設定によって画面に残像のような効果を与えることができます。
・*Alpha*で描画した画像が減衰していく度合いを指定します。この値が高いほど画像がゆっくり減衰し、強い残像効果が得られます。
・スクリーン座標(*StartX,StartY*)～(*EndX,EndY*)で指定した領域に、描画した画像を収束させます。この領域をスクリーンの広さよりも大きくとると広がるように、小さくとると縮むように残像が発生します。
・*Alpha*を0にすることでEFFECT命令による効果を無効にします。

・例

```
EFFECT 60,0,0,640,224,0          弱い残像効果が得られます。
EFFECT 110,0,0,640,224,0         強い残像効果が得られます。
EFFECT 110,-8,-4,648,228,0       拡大していく残像効果が得られます。
EFFECT 0,0,0,0,0,0               効果を無効にします。
```

■LINECOL　ラインプリミティブの色の設定

◆書式

LINECOL *LinePrimNo,Red,Green,Blue,Alpha*

項目	内容	備考
LinePrimNo	プリミティブ番号	範囲：1～1000
Red	赤成分	範囲：0～255
Green	緑成分	範囲：0～255
Blue	青成分	範囲：0～255
Alpha	アルファ値	範囲：0～128

◆解説

・ラインプリミティブの色を設定します。
・*LinePrimNo*にLINECOL命令を適用したいラインプリミティブの番号を指定します。
・*Red,Green,Blue*の各値は、0のとき成分が最も弱く、255のとき成分が最も強くなります。
・*Alpha*は、0に近いほど透明に128に近いほど不透明になります。

◆関連項目

LINEDISP (→p.48参照)　LINEOFF (→p.48参照)　LINEPOS (→p.49参照)

・例

```
LINECOL 1,0,255,0,128            ラインプリミティブ番号1のラインの色を緑色に設定します。
LINECOL 2,35,200,40,64           ラインプリミティブ番号2のラインを黄緑色の半透明に設定します。
```

■LINEDISP　ラインプリミティブの表示

◆書式

LINEDISP *LinePrimNo*

項目	内容	備考
LinePrimNo	プリミティブ番号	範囲：1～1000

◆解説

・*LinePrimNo*に指定したラインプリミティブを表示します。

◆関連項目

LINEOFF (→p.48参照)　LINEPOS (→p.49参照)

・例

```
LINEDISP 55
```
ラインプリミティブ番号55のラインを表示します。

■LINEOFF　ラインプリミティブの非表示

◆書式

LINEOFF *LinePrimNo*

項目	内容	備考
LinePrimNo	プリミティブ番号	範囲：1～1000

◆解説

・*LinePrimNo*に指定したラインプリミティブを非表示にします。
・LINEOFF命令を実行後もLINEPOS命令などによる値の変更は更新されます。

◆関連項目

LINECOL (→p.47参照)　LINEDISP (→p.48参照)　LINEPOS (→p.49参照)

・例

```
LINEOFF 10
```
ラインプリミティブ番号10のラインを非表示にします。

■LINEPOS　ラインプリミティブの表示位置の設定

◆書式

LINEPOS *LinePrimNo,StartX,StartY,EndX,EndY,Z*

項目	内容	備考
LinePrimNo	プリミティブ番号	範囲：1～1000
StartX	始点X座標値	範囲：-1728～2367
StartY	始点Y座標値	範囲：-1936～2159
EndX	終点X座標値	範囲：-1728～2367
EndY	終点Y座標値	範囲：-1936～2159
Z	Z値	範囲：0～16777200

◆解説

・ラインプリミティブの表示位置を設定します。
・*LinePrimNo*に表示位置を設定したいラインプリミティブの番号を指定します。
・スクリーン座標値(*StartX*,*StartY*)と(*EndX*,*EndY*)を結ぶ直線を描画します。
・*Z*にラインプリミティブの奥行きを示すZ値を指定します。*Z*が0に近いほど奥に、16777200に近いほど手前に配置します。

◆関連項目

LINEDISP (→p.48参照)　LINEOFF (→p.48参照)

・例

```
LINEPOS 1,0,0,640,224,0
```

ラインプリミティブ番号1のラインを、始点(0,0)から終点(640,224)に引き、画面の最も奥に配置します。

■PALCOPY　パレットデータのコピー

◆書式

PALCOPY *DstPal,SrcPal*

項目	内容	備考
DstPal	コピー先パレット番号	範囲：1～32
SrcPal	コピー元パレット番号	範囲：1～32

◆解説

・パレットデータを他のパレットへコピーします。

◆関連項目

PALLOAD(→p.50参照)

・例

```
PALCOPY 1,5
```

5番のパレットデータを1番のパレットにコピーします。

■PALLOAD　パレットデータの読み込み

◆書式

PALLOAD ［*ディレクトリ名/*］*ファイル名*

◆解説

・PAL形式のパレットデータを指定したディレクトリにあるファイルから読み込みます。
・*ディレクトリ名*を省略すると、"SYSTEM"ディレクトリからファイルを読み込みます。
・PALLOAD命令を実行すると、ファイルに保存されているパレットデータをすべて読み込みます。
・新たにパレットデータをファイルから読み込むと、それまで使用されていたパレットは上書きされます。
　ただし、読み込むファイル内の使われていないパレット番号は、現在のデータが保持されます。

・例

```
PALLOAD "PAL10"
```
"SYSTEM"ディレクトリ内のファイル名"PAL10"に保存されているパレットデータを読み込みます。

■SPRDISP　スプライトプリミティブの表示

◆書式

SPRDISP *SpritePrimNo*

項目	備考
SpritePrimNo	範囲：1～1000

◆解説

・*SpritePrimNo*で指定したスプライトプリミティブを表示します。

◆関連項目

SPROFF (→p.50参照)　SPRPAL (→p.51参照)　SPRPOS (→p.52参照)
SPRPOS2 (→p.53参照)　SPRTEX (→p.54参照)

・例

```
SPRDISP 66
```
スプライトプリミティブ番号66のスプライトを表示します。

■SPROFF　スプライトプリミティブの非表示

◆書式

SPROFF *SpritePrimNo*

項目	内容	備考
SpritePrimNo	プリミティブ番号	範囲：1～1000

◆解説

・*SpritePrimNo*に指定したスプライトプリミティブを非表示にします。
・SPROFF命令を実行後もSPRPOS命令などによる値の変更は更新されます。

◆関連項目

SPRDISP (→p.50参照)　SPRPAL (→p.51参照)　SPRPOS (→p.52参照)
SPRPOS2 (→p.53参照)　SPRTEX (→p.54参照)

・例

```
SPROFF 66
```
スプライトプリミティブ番号66のスプライトを非表示にします。

■SPRPAL　スプライトプリミティブのパレット設定

◆書式

SPRPAL *SpritePrimNo,PaletteNo,PaletteOfs,AlphaMode*

項目	内容	備考
SpritePrimNo	プリミティブ番号	範囲：1～1000
PaletteNo	パレット番号	範囲：1～32
PaletteOfs	パレットオフセット値	範囲：0～15
AlphaMode	アルファモード	モード：0:減算モード 1:通常モード 2:加算モード

◆解説

・*SpritePrimNo*のスプライトプリミティブが使用するパレットを設定します。
・*PaletteNo*にスプライトプリミティブに適用するパレットの番号を指定します。
・使用するイメージが16色(4bit)モードによって描かれている場合、*PaletteOfs*で使用しているカラーの位置を指定します。

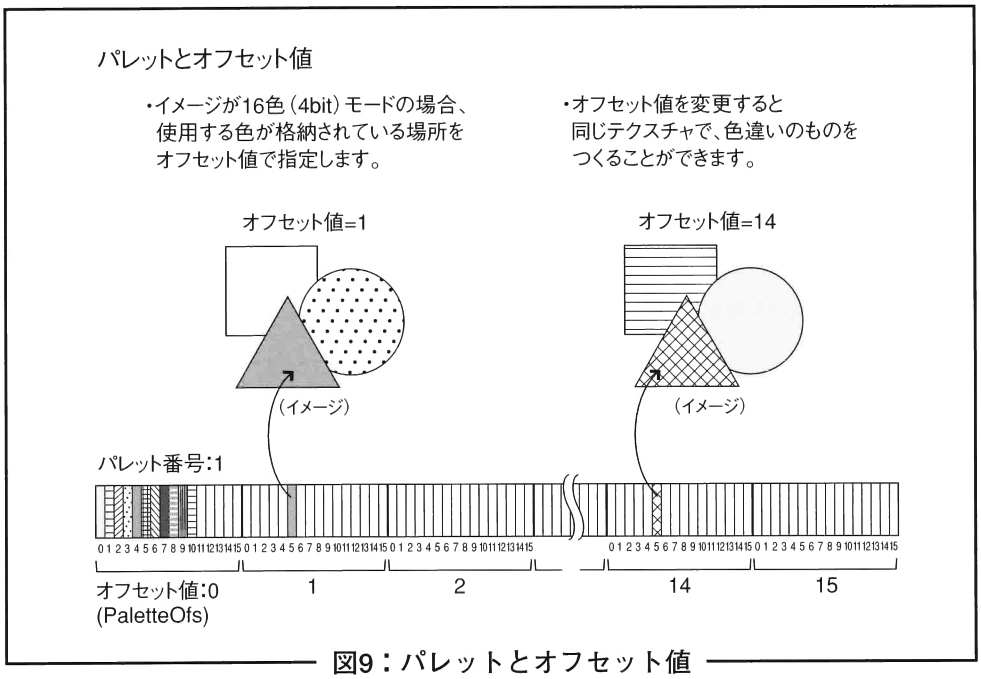
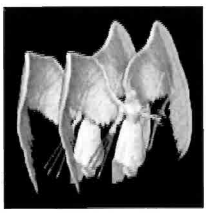
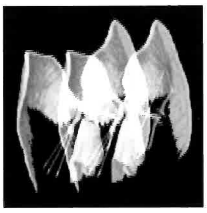


図9：パレットとオフセット値

・使用するイメージが256色(8bit)モードの場合、*PaletteOfs*を0にします。これ以外の値を指定すると適切な色が使用されません。
・*AlphaMode*で半透明色の計算方式を設定します。

減算（AlphaMode=0）


通常（AlphaMode=1）


加算（AlphaMode=2）

◆関連項目

SPRDISP (→p.50参照)　SPROFF (→p.50参照)　SPRPOS (→p.52参照)
SPRPOS2 (→p.53参照)　SPRTEX (→p.54参照)

・例

```
SPRPAL 1,5,3,1
```

スプライトプリミティブ番号1のスプライトに、5番パレットのパレットオフセット値3の色を、通常モードで設定します。

■SPRPOS　スプライトプリミティブの表示位置の設定

◆書式

SPRPOS *SpritePrimNo,ScreenX,ScreenY,Z*

項目	内容	備考
SpritePrimNo	プリミティブ番号	範囲：1～1000
ScreenX	スクリーンX座標値	範囲：-1728～2367（※）
ScreenY	スクリーンY座標値	範囲：-1936～2159（※）
Z	奥行き値	範囲：0～16777200

※スプライトの横または縦のサイズを含めての値です。

◆解説

・*SpritePrimNo*に指定したスプライトプリミティブの表示位置を設定します。
・スプライトの大きさは、SPRTEX命令(→p.54参照)によって指定したテクスチャーデータの大きさになります。
・スクリーン座標値(*ScreenX,ScreenY*)で指定した位置を左上として、スプライトプリミティブを配置します。
・*Z*にスプライトプリミティブの奥行きを示すZ値を指定します。*Z*が0に近いほど奥に配置し、16777200に近いほど手前に配置します。
・各値に範囲外の値を指定するとエラーになります。
・SPRDISP命令(→p.50参照)を実行しないと、画面に表示されません。

◆関連項目

SPRDISP (→p.50参照)　SPROFF (→p.50参照)　SPRPAL (→p.51参照)
SPRPOS2 (→p.53参照)　SPRTEX (→p.54参照)

・例／コントローラ入力によってスプライトを移動させる例

```
100 SPRDISP␣1
110 DO
120  X=PAD(1,14)          コントローラの入力から、変数Xおよび
130  Y=PAD(1,13)          変数Yの値を代入します。
140  SPRPOS␣1,X,Y,0       スプライトをスクリーン座標(X,Y)上に配置します。
150  FSYNC␣1
160 LOOP
```

■SPRPOS2　スプライトプリミティブの表示位置の設定

◆書式

SPRPOS2 *SpritePrimNo*,*StartX*,*StartY*,*EndX*,*EndY*,*Z*

項目	内容	備考
SpritePrimNo	プリミティブ番号	範囲：1～1000
StartX	始点X座標値	範囲：-1728～2367
StartY	始点Y座標値	範囲：-1936～2159
EndX	終点X座標値	範囲：-1728～2367
EndY	終点Y座標値	範囲：-1936～2159
Z	奥行き値	範囲：0～16777200

◆解説

・*SpritePrimNo*に指定したスプライトプリミティブの表示位置を設定します。
スプライトプリミティブを配置する際、配置する領域を指定することができます。これによりスプライトの拡大・縮小、反転を行うことができます。
・スプライトプリミティブを、スクリーン座標(*StartX*,StartY)～(*EndX*,*EndY*)で指定する領域におさまるよう拡大、縮小して表示します。
・*StartX*>*EndX*にするとスプライトプリミティブが左右反転して配置されます。*StartY*>*EndY*にするとスプライトプリミティブが上下反転して配置されます。
・*Z*にスプライトプリミティブの奥行きを示すZ値を指定します。*Z*が0に近いほど奥に配置し、16777200に近いほど手前に配置します。
・各値に範囲外の値を指定するとエラーになります。
・SPRDISP命令(→p.50参照)を実行しないと、画面に表示されません。

◆関連項目

SPRDISP (→p.50参照)　SPROFF (→p.50参照)　SPRPAL (→p.51参照)
SPRPOS (→p.52参照)　SPRTEX (→p.54参照)

・例／コントローラ入力によってスプライトを伸縮表示する例

```
100 SPRDISP␣1
110 DO
120   X1=PAD(1,14)                    コントローラの入力から、スプライトを配置する各値を代入します。
130   Y1=PAD(1,13)
140   X2=PAD(1,16)
150   Y2=PAD(1,15)
160   SPRPOS2␣1,X1,Y1,X2,Y2,0         スプライトを指定した位置に配置します。
170   PRINT␣X1,Y1,X2,Y2               配置位置を表示して確認します。
180   FSYNC␣1
190 LOOP
```

■SPRTEX　スプライトプリミティブのテクスチャー情報設定

◆書式

SPRTEX *SpritePrimNo,TexPageNo,StartTX,StartTY,EndTX,EndTY*

項目	内容	備考
SpritePrimNo	プリミティブ番号	範囲：1～1000
TexPageNo	テクスチャーページ番号	範囲：1～16
StartTX	始点テクスチャーX座標値	範囲：0～512（16色(4bit)モード）、0～256（256色(8bit)モード）
StartTY	始点テクスチャーY座標値	範囲：0～256
EndTX	終点テクスチャーX座標値	範囲：0～512（16色(4bit)モード）、0～256（256色(8bit)モード）
EndTY	終点テクスチャーY座標値	範囲：0～256

◆解説

・イメージデータをスプライトプリミティブとして使用できるように設定します。
・*TexPageNo*に使用するテクスチャーページの番号を指定します。
　テクスチャーページに使用するデータは、TEXLOAD命令(→p.55参照)で読み込みます。
・テクスチャー座標(*StartTX,StartTY*)～(*EndTX,EndTY*)で指定した矩形領域を、スプライトプリミティブとして設定します。

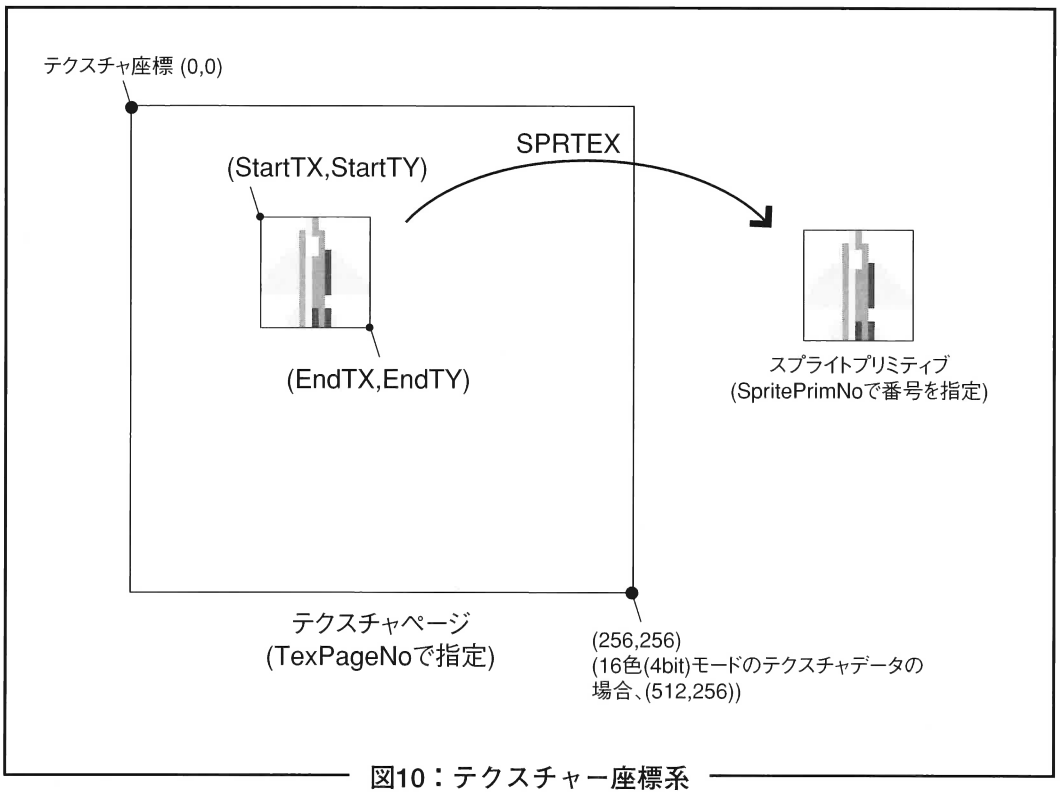


図10：テクスチャー座標系

◆関連項目

SPRDISP (→p.50参照)　SPROFF (→p.50参照)　SPRPAL (→p.51参照)
SPRPOS (→p.52参照)　SPRPOS2 (→p.53参照)

・例

```
SPRTEX 1,2,0,16,64,32
```

1番のスプライトプリミティブに、テクスチャーページ2番上のテクスチャー座標始点(0,16)から終点(64,32)までの矩形領域を割り当てます。

■TEXCOPY　テクスチャーページのコピー

◆書式

TEXCOPY *DstPage,SrcPage*

項目	内容	備考
DstPage	コピー先テクスチャページ番号	範囲：1～16
SrcPage	コピー元テクスチャページ番号	範囲：1～16

◆解説

・テクスチャーページの内容を他のテクスチャーページにコピーします。

◆関連項目

TEXLOAD(→p.55参照)

・例

```
TEXCOPY 1,5
```

テクスチャページ5番の内容をテクスチャーページ1番にコピーします。

■TEXLOAD　テクスチャーデータの読み込み

◆書式

TEXLOAD ［*ディレクトリ名/*］*ファイル名*

◆解説

・*ファイル名*に指定したTEX形式のファイルからテクスチャーデータを読み込み、テクスチャーページに格納します。
・*ディレクトリ名*を指定すると、そのディレクトリ内のファイルからデータを読み込みます。
　*ディレクトリ名*を省略すると"SYSTEM"ディレクトリになります。
・テクスチャーページは16ページから構成され、先頭から順番に1～16までのページ番号が割り当てられます。
・新たにファイルからテクスチャーデータを読み込むと、それまで使用されていたテクスチャーデータが上書きされます。
　ただし、読み込むテクスチャーデータのないテクスチャーページは、現在のデータが保持されます。

◆関連項目

SPRTEX (→p.54参照)

・例

```
TEXLOAD "TEX01"
```

"SYSTEM"ディレクトリ内のファイル名"TEX01"から、テクスチャーデータをテクスチャーページに読み込みます。

■TILECOL　タイルプリミティブの色の設定

◆書式

TILECOL *TilePrimNo,Red,Green,Blue,Alpha*

項目	内容	備考
TilePrimNo	プリミティブ番号	範囲：1～1000
Red	赤成分	範囲：0～255
Green	緑成分	範囲：0～255
Blue	青成分	範囲：0～255
Alpha	アルファ値	範囲：0～128

◆解説

・*TilePrimNo*で指定したタイルプリミティブの色を設定します。
・*Red,Green,Blue*の各値は、0のとき成分が最も弱く255のとき成分が最も強くなります。
・*Alpha*は0に近いほど透明に、128に近いほど不透明になります。
・各値に範囲以外の値を指定するとエラーになります。

◆関連項目

TILEDISP (→p.56参照)　TILEOFF (→p.57参照)　TILEPOS (→p.57参照)

・例

```
TILECOL 1,0,255,0,128
TILECOL 4,0,64,32,64
```

タイルプリミティブ番号1のタイルの色を緑色に設定します。
タイルプリミティブ番号4のタイルの色を深緑の半透明に設定します。

■TILEDISP　タイルプリミティブの表示

◆書式

TILEDISP *TilePrimNo*

項目	内容	備考
TilePrimNo	プリミティブ番号	範囲：1～1000

◆解説

・*TilePrimNo*に指定したタイルプリミティブを表示します。

◆関連項目

TILEOFF (→p.57参照)　TILEPOS (→p.57参照)

・例

```
TILEDISP 77
```

タイルプリミティブ番号77のタイルを表示します。

■TILEOFF　タイルプリミティブの非表示

◆書式

TILEOFF *TilePrimNo*

項目	内容	備考
TilePrimNo	プリミティブ番号	範囲：1～1000

◆解説

・*TilePrimNo*に指定したタイルプリミティブを非表示にします。
・TILEOFF命令を実行後もTILEPOS命令などによる値の変更は行われます。

◆関連項目

TILEDISP (→p.56参照)　TILEPOS (→p.57参照)

・例

```
TILEOFF 88
```

タイルプリミティブ番号88のタイルを非表示にします。

■TILEPOS　タイルプリミティブの表示位置の設定

◆書式

TILEPOS *TilePrimNo,StartX,StartY,EndX,EndY,Z*

項目	内容	備考
TilePrimNo	プリミティブ番号	範囲：1～1000
StartX	始点X座標値	範囲：-1728～2367
StartY	始点Y座標値	範囲：-1936～2159
EndX	終点X座標値	範囲：-1728～2367
EndY	終点Y座標値	範囲：-1936～2159
Z	奥行き値	範囲：0～16777200

◆解説

・*TilePrimNo*に指定したタイルプリミティブの表示位置を設定します。
・タイルを配置する領域を、スクリーン座標(*StartX,StartY*)～(*EndX,EndY*)の矩形領域で指定します。
・*Z*にタイルプリミティブの奥行きを示すZ値を指定します。0が最も奥に配置され、16777200が最も手前に配置されます。
・TILEDISP命令(→p.56参照)を実行しないと、画面に表示されません。

◆関連項目

TILEDISP (→p.56参照)　TILEOFF (→p.57参照)

・例

```
TILEPOS 1,0,0,640,224,0
```

1番のタイルプリミティブを、始点(0,0)から終点(640,224)に示される大きさで、画面の一番奥に設定します。

■TRICOL　トライアングルプリミティブの色の設定

◆書式

TRICOL *TrianglePrimNo,Red,Green,Blue,Alpha*

項目	内容	備考
TrianglePrimNo	プリミティブ番号	範囲：1～1000
Red	赤成分	範囲：0～255
Green	緑成分	範囲：0～255
Blue	青成分	範囲：0～255
Alpha	アルファ値	範囲：0～128

◆解説

・*TrianglePrimNo*に指定したトライアングルプリミティブの色を設定する。
・*Red,Green,Blue*の各値は、0のとき成分が最も弱く、255のとき成分が最も強くなります。
・*Alpha*は0に近いほど透明に、128に近いほど不透明になります。
・各値に範囲以外の値を指定するとエラーになります。

◆関連項目

TRICOL3 (→p.59参照)　TRIDISP (→p.56参照)
TRIOFF (→p.60参照)　TRIPOS (→p.60参照)

・例

```
TRICOL 3,255,255,255,128
TRICOL 4,0,64,32,64
```

トライアングルプリミティブ番号3のトライアングルの色を白に設定します。
トライアングルプリミティブ番号4の取りアングルの色を水色の半透明に設定します。

■TRICOL3　トライアングルプリミティブの各頂点色の設定

◆書式

TRICOL3 *TrianglePrimNo,Red1,Green1,Blue1,Alpha1,Red2,Green2,Blue2,Alpha2,Red3,Green3,Blue3,Alpha3*

項目	内容	備考
TrianglePrimNo	プリミティブ番号	範囲：1～1000
Red1	頂点１の赤成分	範囲：0～255
Green1	頂点１の緑成分	範囲：0～255
Blue1	頂点１の青成分	範囲：0～255
Alpha1	頂点１のアルファ値	範囲：0～128
Red2	頂点２の赤成分	範囲：0～255
Green2	頂点２の緑成分	範囲：0～255
Blue2	頂点２の青成分	範囲：0～255
Alpha2	頂点２のアルファ値	範囲：0～128
Red3	頂点３の赤成分	範囲：0～255
Green3	頂点３の緑成分	範囲：0～255
Blue3	頂点３の青成分	範囲：0～255
Alpha3	頂点３のアルファ値	範囲：0～128

◆解説

・*TrianglePrimNo*に指定したトライアングルプリミティブの色を頂点毎に設定します。
　トライアングル内は、各頂点の色のグラデーションで塗りつぶされます。
・各色成分は、0のとき成分が最も弱く、255のとき成分が最も強くなります。
・各アルファ値は0に近いほど透明に、128に近いほど不透明になります。
・各値に範囲以外の値を指定するとエラーになります。

◆関連項目

TRICOL (→p.58参照)　　TRIDISP (→p.59参照)
TRIOFF (→p.60参照)　　TRIPOS (→p.60参照)

・例

```
TRICOL 3,255,255,255,128, 0,255,255,64,255,0,0,34
```

トライアングルプリミティブ番号3のトライアングルの、頂点1の色を白に、頂点2の色を水色の半透明に、頂点3の色を赤で透明度34に設定します。

■TRIDISP　トライアングルプリミティブの表示

◆書式

TRIDISP *TrianglePrimNo*

項目	内容	備考
TrianglePrimNo	プリミティブ番号	範囲：1～1000

◆解説

・*TrianglePrimNo*に指定したトライアングルプリミティブを表示します。

◆関連項目

TRIOFF (→p.60参照)　　TRIPOS (→p.60参照)

・例

```
TRIDISP 99
```

トライアングルプリミティブ番号99のトライアングルを表示します。

■TRIOFF　トライアングルプリミティブの非表示

◆書式

TRIOFF *TrianglePrimNo*

項目	内容	備考
TrianglePrimNo	プリミティブ番号	範囲：1～1000

◆解説

・*TrianglePrimNo*に指定したトライアングルプリミティブを非表示にします。
・TRIOFF命令を実行後もTRIPOS命令などによる値の変更は更新されます。

◆関連項目

TRICOL (→p.58参照)　TRICOL3 (→p.59参照)
TRIDISP (→p.59参照)　TRIPOS (→p.60参照)

・例

```
TRIOFF 101
```

トライアングルプリミティブ番号101のトライアングルを非表示にします。

■TRIPOS　トライアングルプリミティブの表示位置の設定

◆書式

TRIPOS *TrianglePrimNo,ScreenX1,ScreenY1,ScreenX2,ScreenY2,ScreenX3,ScreenY3,Z*

項目	内容	備考
TrianglePrimNo	プリミティブ番号	範囲：1～1000
ScreenX1	頂点１X座標値	範囲：-1728～2367
ScreenY1	頂点１Y座標値	範囲：-1936～2159
ScreenX2	頂点２X座標値	範囲：-1728～2367
ScreenY2	頂点２Y座標値	範囲：-1936～2159
ScreenX3	頂点３X座標値	範囲：-1728～2367
ScreenY3	頂点３Y座標値	範囲：-1936～2159
Z	奥行き値	範囲：0～16777200

◆解説

・*TrianglePrimNo*に指定したトライアングルプリミティブの表示位置を設定します。
・トライアングルの各頂点の座標を、スクリーン上の座標(*ScreenX1,ScreenY1*)、(*ScreenX2,ScreenY2*)、(*ScreenX3,ScreenY3*)で指定します。
・*Z*にトライアングルプリミティブの奥行きを示すZ値を指定します。

◆関連項目

TRIDISP (→p.59参照)　TRIOFF (→p.60参照)

・例

```
TRIPOS 1,0,0,320,224,640,0,0
```

トライアングルプリミティブ番号1のトライアングルを、各頂点の座標値(0,0)(320,224)(640,0)で、画面の最も奥に配置するよう設定します。

3Dグラフィックス関連

■3次元空間の表現について

◆モデルの表示

・モデルは仮想的に設定された3次元空間内に配置されます。
・仮想的に用意されたカメラによって撮影した映像が、スクリーンに表示されます。

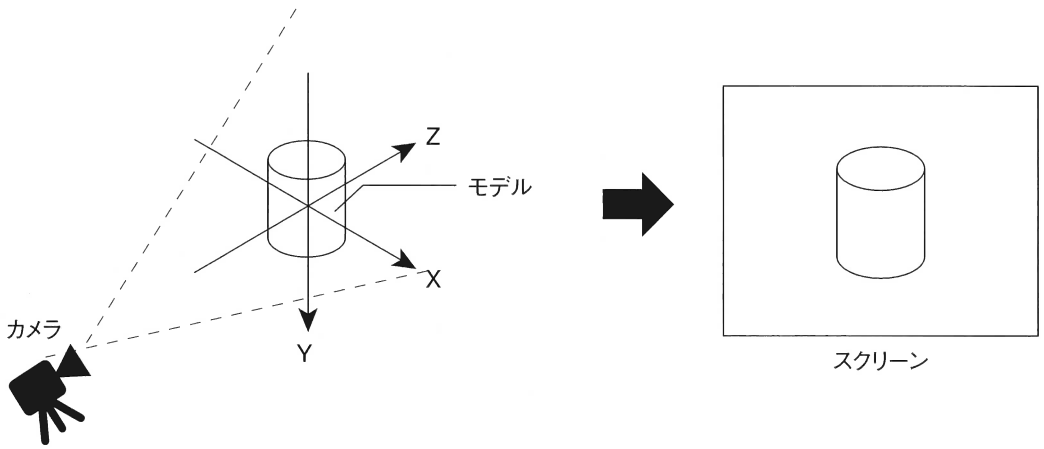


図11：モデルの表示

・モデルを配置する位置情報は、ワールド空間座標系を用いて指定します。

◆ワールド空間

・仮想的に設定された3次元空間です。モデルおよびカメラはこの空間内に配置されます。
・ワールド空間内のモデルは、ワールド座標（x,y,z）によって位置を表します。
・ワールド空間は、すべてのモデルに共通する絶対的な空間です。

◆ローカル空間

・モデル固有の座標系によって表される空間です。
・モデルの回転はローカル座標軸を基準にします。

◆親モデルと子モデル

・ふたつのモデル間に主従関係を設定することができます。一方のモデルに対し従属するモデルを**子モデル**、従属する対象を**親モデル**と呼びます。
・親モデルの移動および回転は、従属する子モデルにも反映されます。

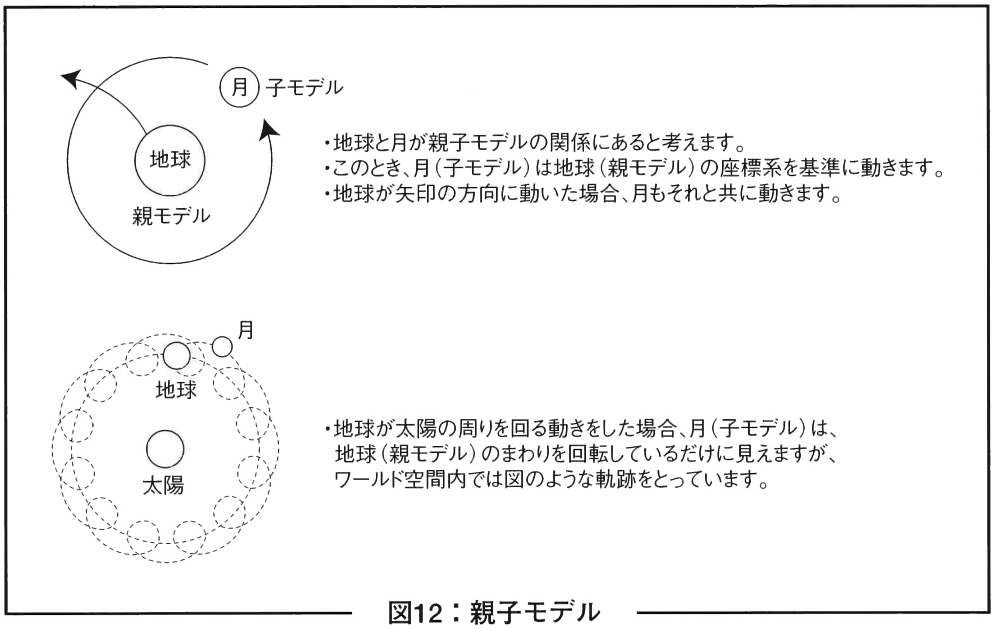


図12：親子モデル

■モデル

◆モデルデータ

・"EZ-BASIC"で扱うポリゴンモデルです。モデルデータは、モデラー"POLY-CRAFT"で作成します。内容に応じてテクスチャーおよびパレットデータを伴います。
・モデルデータをモデルバッファに読み込んだ後、モデル番号にモデルバッファ内のモデルデータを結び付けます。
・モデル番号は、プログラムでモデルデータを扱うための管理番号です。番号は1～1000の範囲で指定し最大1000個まで使用することができます。
・ファイルからモデルデータを読み込むと、モデルバッファに格納されます。モデルバッファは8個あり、各モデルバッファ内に最大で10個のモデルが格納されています。
モデルバッファ番号は先頭から1～8までの番号によって管理され、モデルバッファ内のモデルは先頭から1～10の番号によって管理されます。

◆当たり判定

・モデルデータに設定された当たり判定用の設定を用いて、プログラム上でモデルどうしの当たり判定を行うことができます。
当たり判定には、接触の有無、接触しているモデル間の距離があります。
・当たり判定を行うためには、"POLY-CRAFT"上で当たり判定用の設定をしておく必要があります。
・モデルの当たり判定にはGETCOLL命令(→p.70参照)またはCOLLISION命令(→p.101参照)を用います。

◆Zソート

・モデルをZ値(→p.44参照)によってソート(並び替え)します。
・Zソートの設定はZSORT命令(→p.81参照)によって行います。
・Zバッファ問題(→p.44参照)を軽減することができます。

■カメラ

◆カメラの機能

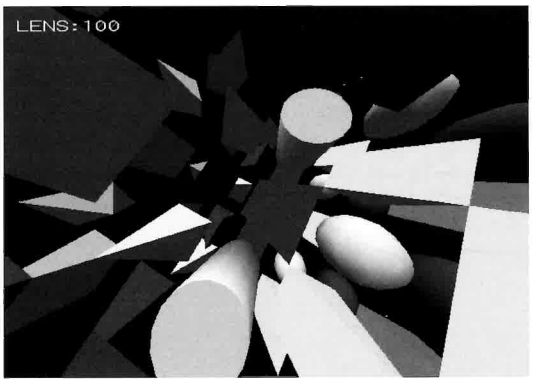
・3次元空間内を撮影し、その映像をスクリーンに描画することができる仮想的なカメラです。
・カメラはモデルと同様に、ワールド空間内に配置し向きを指定します。
カメラ自体は見ることのできない仮想的な存在です。
・最大ふたつのカメラを扱うことができます。

◆スクリーン分割

・スクリーンを左右または上下に分割して表示を行うことができます。また、通常の1画面表示上の任意の位置にもうひとつの分割画面を表示することができます。
・分割された一方のスクリーンにカメラ1の映像が投影され、他方のスクリーンにカメラ2の映像が投影されます。

◆レンズ

・カメラのレンズの性質を設定することができます。この値を設定することで広角や望遠といった映像を得ることができます。



■光源

◆ライト

・ワールド空間内にライトを配置することができます。ライトを配置することによって、モデルに光を照らしたような効果を与えることができます。
・ライトは最大3つを扱うことができます。各ライトは1～3のライト番号によって管理します。
・ライトは平行光源です。
・ライトは色および向きを設定することができます。

◆環境光

・アンビエントカラーです。
・ライトと異なり向きはありません。ワールド空間内すべてのモデルに対して環境光の効果を与えます。

■フォグ

◆フォグの機能

・3次元空間内に擬似的な霧(フォグ)のような効果を与えます。
・フォグの機能によってカメラから遠方になるほどかすんだ効果が得られます。

◆フォグの設定

・フォグの設定には、フォグがかかりはじめる距離とフォグが完全にかかる距離があります。
このふたつの値を設定することによって、この2点間で徐々にフォグがかかるような画像を生成します。
フォグがかかりはじめる距離を**NEAR値**、フォグが完全にかかる距離を**FAR値**といいます。
・またフォグの色を設定することができます。

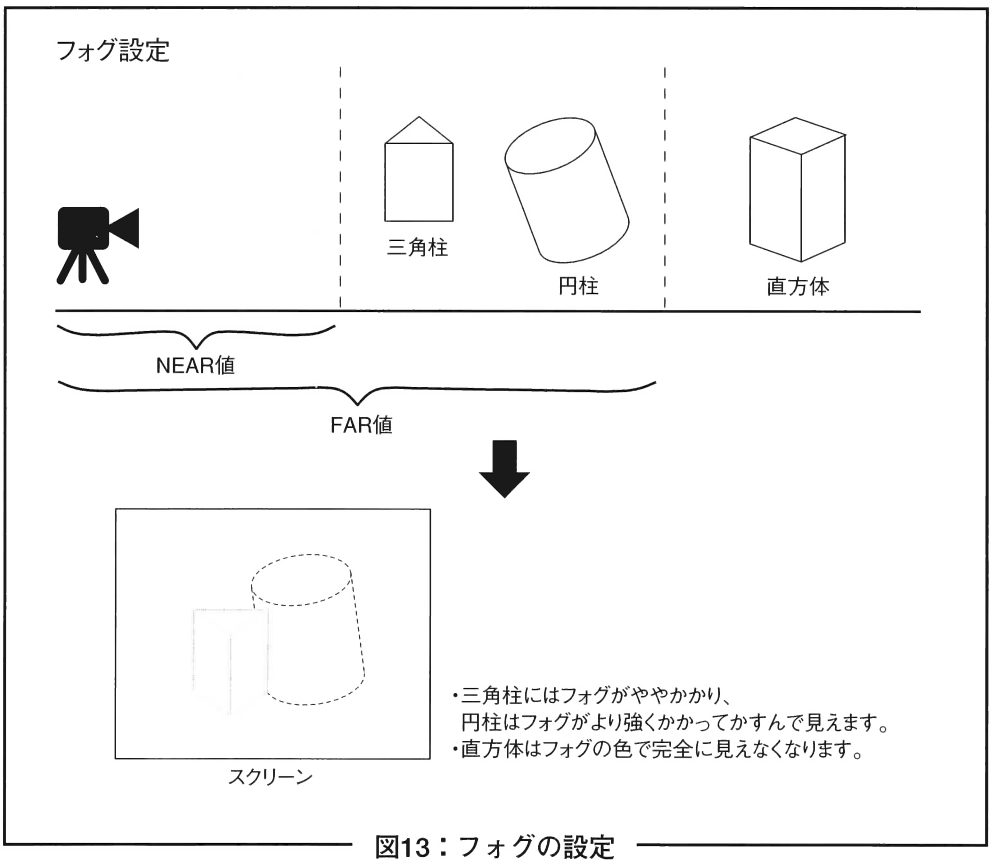


図13：フォグの設定

■ALTEST　アルファ値によるZ値書き込みの制御

◆書式

ALTEST *アルファ値*

◆解説

・指定した*アルファ値*以下のピクセルはZ値を書き込みません。
・RGB各値には影響がありません。

・例

```
ALTEST 64
```
アルファ値が64以下のピクセルはZ値が書き込まれません。

■AMBCOL　環境光の設定

◆書式

AMBCOL *Red,Green,Blue*

項目	内容	備考
Red	赤成分	範囲：0～255
Green	緑成分	範囲：0～255
Blue	青成分	範囲：0～255

◆解説

・ワールド空間内の環境光を*Red,Green,Blue*で指定した色に設定します。
・ワールド空間内のオブジェクトに設定した環境光が反映されます。テキストおよびプリミティブには反映されません。

◆関連項目

LIGCOL (→p.74参照)

・例

```
AMBCOL 0,0,255
```
環境光が青になります。

■CAMLENS　カメラの画角設定

◆書式

CAMLENS *CameraNo,Lens*

項目	内容	備考
CameraNo	カメラ番号	範囲：1～2
Lens	レンズ値	範囲：100～10000 初期値：512

◆解説

・*CameraNo*で指定したカメラの画角を設定します。
・*Lens*に画角と対応した値を指定します。
　この値を大きくすると望遠レンズのような効果が得られ、値を小さくすると広角レンズのような効果を得ることができます。

・例

```
CAMLENS 1,2000
```

カメラ番号1のカメラを望遠レンズのような効果に設定します。

■CAMPOS　カメラの位置設定

◆書式

CAMPOS *CameraNo,WorldX,WorldY,WorldZ*

項目	内容	備考
CameraNo	カメラ番号	範囲：1～2
WorldX	ワールドX座標値	
WorldY	ワールドY座標値	
WorldZ	ワールドZ座標値	

◆解説

・*CameraNo*で指定したカメラの位置を設定します。
・ワールド座標値(*WorldX,WorldY,WorldZ*)で指定した位置にカメラを配置します。
・カメラ1の初期位置はワールド座標(0,0,-100)、カメラ2の初期位置はワールド座標(0,0,100)です。

◆関連項目

CAMVEC (→p.68参照)　CAMROT (→p.67参照)

・例

```
CAMPOS 1,0,0,0
```

カメラ番号1のカメラをワールド空間内の原点(0,0,0)にセットします。

■CAMROT　カメラの向きの設定

◆書式

CAMROT *CameraNo,RotX,RotY,RotZ*

項目	内容	備考
CameraNo	カメラ番号	範囲：1～2
RotX	X軸角度	単位：ラジアン
RotY	Y軸角度	単位：ラジアン
RotZ	Z軸角度	単位：ラジアン

◆解説

・*CameraNo*で指定したカメラの向きを各軸の角度により設定します。
・*RotX,RotY,RotZ*に、各軸の角度を指定します。

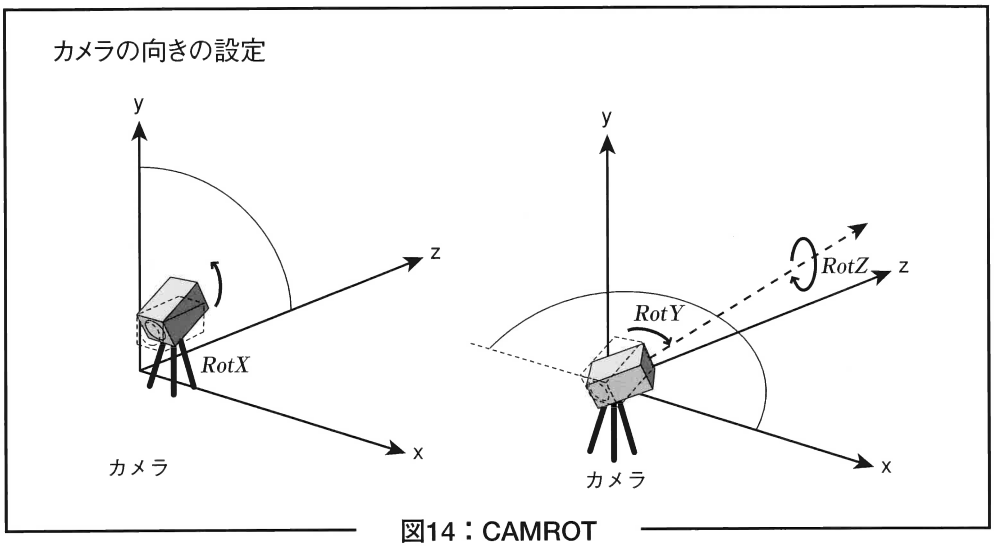


図14：CAMROT

・単位にはラジアンを用います。角度をラジアンに変換するには、RAD関数(→p.98参照)を用います。

◆関連項目

CAMPOS (→p.66参照)　　CAMVEC (→p.68参照)

・例1

```
CAMROT 1,PI/2,0,0
CAMROT 1,0,0,RAD(90)
```

カメラ番号1のカメラを真下に向けます。
カメラ番号1のカメラを左に90度傾けます。

・例2

```
100 MODDISP␣1
110 DO
120  X=X+RAD(PAD(1,3)-PAD(1,4))/80
130  Y=Y+RAD(PAD(1,2)-PAD(1,1))/80
140  Z=Z+RAD(PAD(1,10)-PAD(1,9))/80
150  CAMROT␣1,X,Y,Z
160  PRINT␣IP(DEG(X)),IP(DEG(Y)),IP(DEG(Z))
170  FSYNC␣1
180 LOOP
190 END
```

方向キー上・下でカメラをX軸回転させます。
方向キー左・右でカメラをY軸回転させます。
L1ボタン,R1ボタンでカメラをZ軸回転させます。

■CAMVEC　カメラの向きの設定

◆書式

CAMVEC *CameraNo,VectorX,VectorY,VectorZ*

項目	内容	備考
CameraNo	カメラ番号	範囲：1～2
VectorX	ベクトルX	
VectorY	ベクトルY	
VectorZ	ベクトルZ	

◆解説

・*CameraNo*に指定したカメラの向きをベクトルによって設定します。
・*VectorX*、*VectorY*、*VectorZ*を用いて、カメラの向きを指定します。

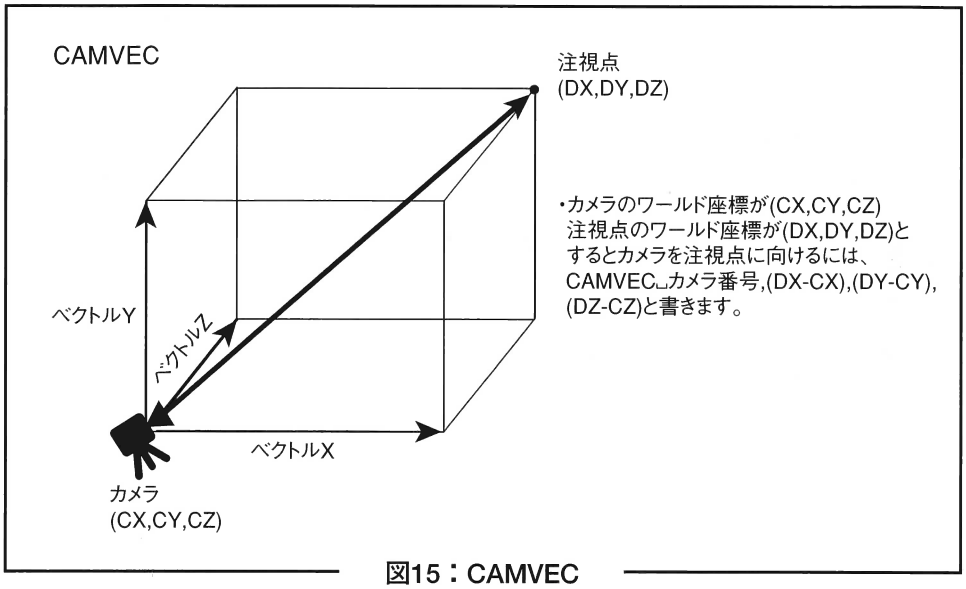


図15：CAMVEC

◆関連項目

CAMPOS (→p.66参照)　　CAMROT (→p.67参照)

・例

```
100 CX,CY=0                          カメラ1のワールド座標用変数
110 CZ=100
120 MODDISP␣1
130 MODDISP␣2
140 DO
150  S=S+1
160  DX=100*SIN(RAD(S))              モデル1のワールド座標用変数
170  DY,DZ=0
180  MODPOS␣1,DX,DY,DZ               モデル1を配置
190  CAMPOS␣1,CX,CY,CZ               カメラ1を配置
200  CAMVEC␣1,DX-CX,DY-CY,DZ-CZ      カメラ1はモデル1に注目
210  FSYNC␣1
220 LOOP
230 END
```

■FOGCOL　霧効果の色の設定

◆書式

FOGCOL *Red,Green,Blue*

項目	内容	備考
Red	赤成分	範囲：0～255
Green	緑成分	範囲：0～255
Blue	青成分	範囲：0～255

◆解説

・ワールド空間のフォグカラー(霧効果の色)を設定します。
・*Red,Green,Blue*に各色成分の値を指定します。
・霧効果はモデルのみ影響します。プリミティブおよびテキストには影響ありません。

◆関連項目

FOGSET (→p.69参照)

・例

```
FOGCOL 255,255,255
```

フォグカラーを白に設定します。

■FOGSET　霧効果の範囲設定

◆書式

FOGSET *Near,Far*

項目	内容	備考
Near	フォグ開始距離	範囲：0～10000000
Far	フォグ完成距離	範囲：0～10000000

◆解説

・フォグ(霧)の効果範囲を設定します。
・効果範囲は、カメラからの距離で指定します。
*Near*にフォグがかかりはじめる距離を指定し、*Far*にフォグが完全にかかる距離を指定します。

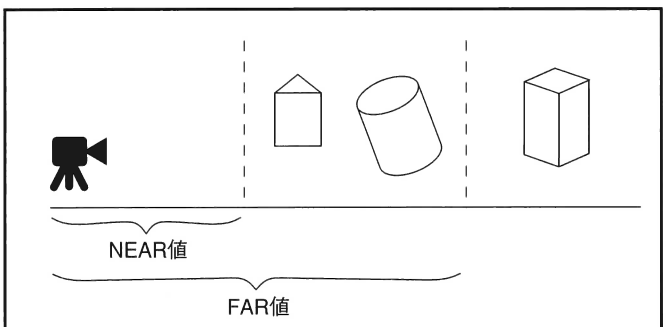


図16：Near値とFar値

・*Near*>=*Far*の場合、エラーになります。
・霧効果はモデルにのみ反映されます。テキストおよびプリミティブには反映されません。

◆関連項目

FOGCOL (→p.69参照)

・例

```
FOGSET 20,100
```

ワールド空間で、カメラからの距離が20の位置からフォグ効果を開始し、カメラからの距離が100の位置で完全にフォグの色にします。

■GETCOLL　モデル間の衝突距離の取得

◆書式

GETCOLL *VarModel,VarX,VarY,VarZ,ModelSrc,ModelStart,ModelEnd*

項目	内容	備考
VarModel	取得用変数名(モデル番号)	
VarX	取得用変数名(ベクトルX値)	
VarY	取得用変数名(ベクトルY値)	
VarZ	取得用変数名(ベクトルZ値)	
ModelSrc	当たり判定モデル番号	
ModelStart	判定開始モデル番号	
ModelEnd	判定終了モデル番号	

◆解説

・*ModelSrc*に指定したモデルに対し、モデル番号*ModelStart*～*ModelEnd*までのモデルと当たり判定を行い、衝突しているモデル間の距離を返します。
　接触しているモデルがあった場合、そのモデルと*ModelSrc*番のモデルとのベクトルを、各取得用変数に代入します。
・代入されるベクトルは、各モデルが持つ当たり判定用領域の中心からのベクトルによって求められます。
・接触しているモデルがない場合、*VarModel*に0が代入されます。
・モデルデータ内に当たり判定用の設定をしておく必要があります。

◆関連項目

COLLISION(→p.101参照)

・例

```
GETCOLL N,X,Y,Z,1,10,20
```

1番モデルに対し10番モデルから20番モデルまでの当たり判定を行い、衝突しているモデル番号を変数Nに、1番モデルの衝突領域中心から衝突しているモデルの衝突領域中心までのベクトルを変数X,Y,Zに得ます。

■GETLPOS　ローカル座標値の取得

◆書式

GETLPOS *LocalX,LocalY,LocalZ,ModelNo,WorldX,WorldY,WorldZ*

項目	内容	備考
LocalX	取得用変数名(ローカルX座標値)	
LocalY	取得用変数名(ローカルY座標値)	
LocalZ	取得用変数名(ローカルZ座標値)	
ModelNo	モデル番号	範囲：1～1000
WorldX	ワールドX座標値	
WorldY	ワールドY座標値	
WorldZ	ワールドZ座標値	

◆解説

・指定したワールド座標値(*WorldX*, *WorldY*, *WorldZ*)の、*ModelNo*で指定したモデルのローカル座標上の位置を取得します。

◆関連項目

GETWPOS (→p.73参照)　GETSPOS (→p.72参照)

・例

```
GETLPOS A,B,C,5,100,50,20
```

ワールド座標(100,50,20)の、5番モデルにおけるローカル座標値を変数A,B,Cに取得します。

■GETSPOS　スクリーン座標値の取得

◆書式

GETSPOS *ScreenX,ScreenY,ScreenZ, CameraNo, WorldX,WorldY,WorldZ*

項目	内容	備考
ScreenX	取得用変数名(スクリーンX座標値)	
ScreenY	取得用変数名(スクリーンY座標値)	
ScreenZ	取得用変数名(スクリーンZ座標値)	
CameraNo	カメラ番号	範囲：1～2
WorldX	ワールドX座標値	
WorldY	ワールドY座標値	
WorldZ	ワールドZ座標値	

◆解説

・ワールド座標値(*WorldX,WorldY,WorldZ*)で指定した位置の、スクリーン上の座標値を求めます。
・*CameraNo*でカメラを指定します。
・スクリーン座標値は取得用変数(*ScreenX,ScreenY,ScreenZ*)に代入されます。

◆関連項目

GETLPOS (→p.71参照)　GETWPOS (→p.73参照)

・例1

```
GETSPOS SX,SY,SZ,1,0,0,0
```

ワールド空間座標値(0,0,0)の位置が、カメラ番号1のカメラから見てスクリーン座標上にある位置を、スクリーン座標値(SX,SY,SZ)に取得します。

・例2

```
100 TEXTMODE␣1
110 MODDISP␣1
120 DO
130  X=(PAD(1,14)-128)/8          左スティックでモデルのワールドX座標値を変更
140  Y=(PAD(1,13)-128)/8          左スティックでモデルのワールドY座標値を変更
150  Z=PAD(1,3)                   方向キー上でモデルのワールドZ座標値を変更
160  GETSPOS␣SX,SY,SZ,1,X,Y,Z     ワールド空間座標値とカメラ番号からスクリーン座標値を取得
170  MODPOS␣1,X,Y,Z
180  CLEAR
190  LOCATE␣SX/23,SY/10           取得したスクリーン座標値をもとにテキスト描画位置を指定
200  PRINT␣INT(SX),INT(SY),INT(SZ)  現在のスクリーン座標値を整数部のみ表示
210  FSYNC␣1
220 LOOP
230 END
```

■GETWPOS　ワールド座標値の取得

◆書式

GETWPOS *WorldX,WorldY,WorldZ, ModelNo, LocalX,LocalY,LocalZ*

項目	内容	備考
WorldX	取得用変数名(ワールドX座標値)	
WorldY	取得用変数名(ワールドY座標値)	
WorldZ	取得用変数名(ワールドZ座標値)	
ModelNo	モデル番号	範囲：1～1000
LocalX	ローカルX座標	
LocalY	ローカルY座標	
LocalZ	ローカルZ座標	

◆解説

・*ModelNo*で指定したモデルのローカル座標値(*LocalX,LocalY,LocalZ*)の、ワールド座標上の位置を取得します。
・ワールド座標値は取得用変数(*WorldX,WorldY,WorldZ*)に代入されます。

◆関連項目

GETLPOS (→p.71参照)　GETSPOS (→p.72参照)

・例

```
100 MODDISP␣1
110 MODDISP␣2
120 DO
130  A=A+(PAD(1,1)-PAD(1,2))/100          1番モデルをコントローラの方向キー左・右で操作します。
140  MODPOS␣1,A,0,100
150  IF␣PAD(1,8)>0␣THEN                  □ボタンが押されている間､160行を実行します。
160   GETWPOS␣X,Y,Z,1,0,20,0              1番モデルのローカル座標(0,20,0)のワールド座標を求めます。
170  END␣IF
180  MODPOS␣2,X,Y,Z                       ワールド座標(X,Y,Z)に2番モデルを配置します。
190  FSYNC␣1
200 LOOP
210 END
```

■LIGCOL　ライトの色の設定

◆書式

LIGCOL *LightNo,Red,Green,Blue*

項目	内容	備考
LightNo	ライト番号	範囲：1～3
Red	赤成分	範囲：0～255
Green	緑成分	範囲：0～255
Blue	青成分	範囲：0～255

◆解説

・*LightNo*で指定したライトの色を設定します。
　ライトによってワールド空間のモデルを照らします。
　ライトとして3つの平行光源があり、それぞれの色を設定することができます。
・*Red,Green,Blue*に各色成分の値を指定します。
・ライトはモデルのみ影響します。プリミティブおよびテキストには影響ありません。

◆関連項目

LIGVEC (→p.75参照)

・例／コントローラ入力によってライトの色を変更する例

```
100 MODDISP␣1
110 DO
120  R=PAD(1,8)          赤成分を□ボタンの入力量で決定
130  G=PAD(1,5)          緑成分を△ボタンの入力量で決定
140  B=PAD(1,6)          青成分を○ボタンの入力量で決定
150  LIGCOL␣1,R,G,B      ライトの色を変数R,G,Bの値に変更します。
160  PRINT␣R,G,B
170 LOOP
180 END
```

■LIGVEC　ライトの方向設定

◆書式

LIGVEC *LightNo,VectorX,VectorY,VectorZ*

項目	内容	備考
LightNo	ライト番号	範囲：1～3
VectorX	ベクトルX	
VectorY	ベクトルY	
VectorZ	ベクトルZ	

◆解説

・*LightNo*で指定したライトの、照らす方向を設定します。
・*VectorX,VectorY,VectorZ*で、ライトを照らす方向を指定します。

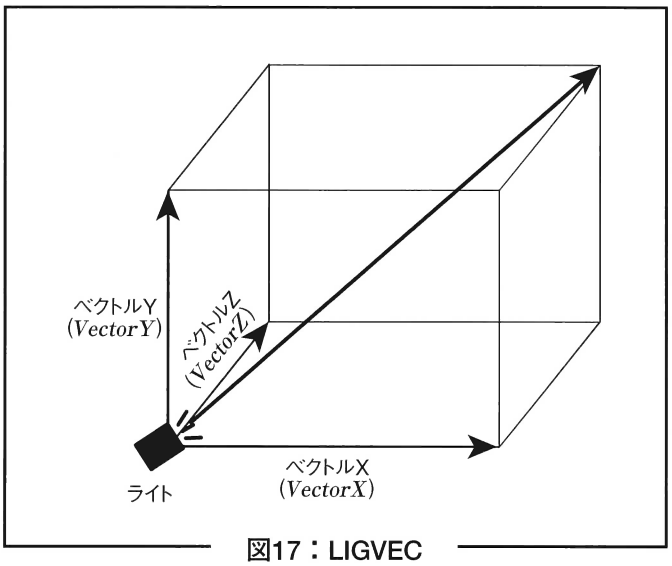


図17：LIGVEC

◆関連項目

LIGCOL (→p.74参照)

・例

```
100 MODDISP␣1
110 DO
120  X=PAD(1,8)-128
130  Y=PAD(1,5)-128
140  Z=PAD(1,6)-128
150  LIGVEC␣1,X,Y,Z
160  PRINT␣X,Y,Z
170 LOOP
180 END
```

ベクトルXの成分を□ボタンの入力量で決定
ベクトルYの成分を△ボタンの入力量で決定
ベクトルZの成分を○ボタンの入力量で決定
ベクトルX,Y,Zの値によりライトの方向を設定します

■MODDISP　モデルの表示

◆書式

MODDISP *ModelNo*

項目	内容	備考
ModelNo	モデル番号	範囲：1～1000

◆解説

・*ModelNo*に指定したモデルを表示します。

◆関連項目

MODJOIN (→p.77参照)　MODOFF (→p.78参照)　MODOPT (→p.79参照)
MODPOS (→p.79参照)　MODSET (→p.81参照)　MODROT (→p.80参照)

・例／モデルを表示する例

```
100 FOR␣A=1␣TO␣10
110  MODDISP␣A
120  MODPOS␣A,(A-5)*10,0,0
130 NEXT␣A
140 END
```

繰り返しを用いてモデルを一斉に表示します。
変数Aの値が指すモデルを表示します。

■MODJOIN　モデルと親モデルとの結合

◆書式

MODJOIN *ModelNo,ParentModelNo*

項目	内容	備考
ModelNo	モデル番号	範囲：1～1000
ParentModelNo	親モデル番号	範囲：0～1000

◆解説

・*ModelNo*で指定したモデルを、*ParentModelNo*で指定したモデルと結合します。
・*ParentModelNo*と*ModelNo*のモデルは親子関係となり、子モデルの*ModelNo*番のモデルは親モデルのローカル座標系を基準にします。
・指定する値は*ParentModelNo*<*ModelNo*でなければなりません。
・*ParentModelNo*に0を指定すると、結合された子モデルが解除されます。

◆関連項目

MODDISP (→p.76参照)　MODOFF (→p.78参照)　MODOPT (→p.79参照)
MODPOS (→p.79参照)　MODSET (→p.81参照)　MODROT (→p.80参照)

・例／モデルを、親モデルに接続・解除する例

```
100 MODDISP␣1                          親モデル
110 MODDISP␣2                          子モデル
120 DO
130  X=(PAD(1,14)-128)/6
140  Y=(PAD(1,13)-128)/6
150  IF␣PAD(1,8)>0␣THEN MODJOIN␣2,1    □ボタンを押すと親モデルに接続
160  IF␣PAD(1,7)>0␣THEN MODJOIN␣2,0    ×ボタンを押すと親モデルから解除
170  MODPOS␣1,X,Y,0                    親モデルをコントローラにより移動します
180  R=R+1                             子モデルの回転角度の更新
190  MODROT␣2,0,RAD(R),0
200  MODPOS␣2,10,0,0                   親との接続時は親のローカル座標系上、解除時はワールド座標系を対象としてMODPOS命令が実行されます。
210  FSYNC␣1
220 LOOP
230 END
```

＜解説＞

150 モデル番号1番のモデルと2番のモデルを親子関係にします。
　これにより、190～200行のモデルの配置に関する命令はすべて親モデルのローカル座標系を基準にします。
　親モデルを動かすと、子モデルも動きます。
160 親子関係を削除します。
　これにより、190～200行の命令はすべてワールド座標系を基準にします。
　親モデルであった1番のモデルが動いても、子モデルであった2番のモデルは影響されません。

■MODLOAD　モデルデータの読み込み

◆書式

MODLOAD ［*ディレクトリ名/*］*ファイル名,モデルバッファ番号*

項目	備考
モデルバッファ番号	範囲：1～8

◆解説

・*ファイル名*で指定したファイルからMOD形式のモデルデータを読み込み、*モデルバッファ番号*で指定したモデルバッファに格納します。
・*ディレクトリ名*を指定するとそのディレクトリ内のファイルから読み込みます。*ディレクトリ名*を省略すると、"SYSTEM"ディレクトリからファイルを読み込みます。
・モデルバッファは8つあります。モデルバッファひとつにつき最大10種類のモデルが格納されます。

◆関連項目

MODSET (→p.81参照)　MODDISP (→p.76参照)

・例

```
MODLOAD "TEST",1
MODLOAD FNAM$,3
```

ファイル"TEST.MOD"内のデータをモデルバッファ1番に読み込みます。
文字列変数FNAM$の値が示すファイル名のデータをモデルバッファ3番に読み込みます。

■MODOFF　モデルの非表示

◆書式

MODOFF *ModelNo*

項目	内容	備考
ModelNo	モデル番号	範囲：1～1000

◆解説

・*ModelNo*に指定したモデルを非表示にします。

◆関連項目

MODDISP (→p.76参照)

・例

```
MODOFF 123
```

モデル番号123番のモデルを非表示にします。

■MODOPT　モデルのオプション設定

◆書式

MODOPT *ModelNo,Scale,Light,Fog,Alpha*

項目	内容	備考
ModelNo	モデル番号	範囲：1～1000
Scale	拡大率	範囲：0.001～1000　原寸:1
Light	光源計算	範囲：0:光源非適用 1:光源適用
Fog	フォグ効果	範囲：0:フォグ非適用 1:フォグ適用
Alpha	アルファモード	モード：0:減算モード 1:通常モード 2:加算モード

◆解説

・*ModelNo*で指定したモデルのオプション設定を行います。
・モデルの拡大率および光源計算の有無、フォグ適用の有無、アルファモードの設定を行います。

・例

```
MODOPT 1,2,0,1,1
```

1番のモデルを2倍の大きさにし、ライトを適用しないようにします。

■MODPOS　モデルの配置

◆書式

MODPOS *ModelNo,WorldX,WorldY,WorldZ*

項目	内容	備考
ModelNo	モデル番号	範囲：1～1000
WorldX	ワールドX座標値	
WorldY	ワールドY座標値	
WorldZ	ワールドZ座標値	

◆解説

・*ModelNo*で指定したモデルを配置するワールド座標を設定します。
・ワールド座標値(*WorldX,WorldY,WorldZ*)に配置する座標値を指定します。

◆関連項目

MODDISP (→p.76参照)　MODJOIN (→p.77参照)　MODOPT (→p.79参照)
MODOFF (→p.78参照)　MODROT (→p.80参照)　MODSET (→p.81参照)

・例／MODPOS命令を用いたモデル移動の例

```
100 MODDISP␣1
110 DO
120   X=X+(PAD(1,1)-PAD(1,2))/80          コントローラ入力によって変数Xの値を更新
130   Y=Y+(PAD(1,4)-PAD(1,3))/80          コントローラ入力によって変数Yの値を更新
140   Z=Z+(PAD(1,11)-PAD(1,9))/80         コントローラ入力によって変数Zの値を更新
150   MODPOS␣1,X,Y,Z                      ワールド座標値X,Y,Zに配置
160   FSYNC␣1
170 LOOP
180 END
```

■MODROT　モデルの角度の設定

◆書式

MODROT *ModelNo,RotX,RotY,RotZ*

項目	内容	備考
ModelNo	モデル番号	範囲：1～1000
RotX	X軸角度	単位：ラジアン
RotY	Y軸角度	単位：ラジアン
RotZ	Z軸角度	単位：ラジアン

◆解説

・*ModelNo*で指定したモデルの角度を設定します。
・角度(*RotX,RotY,RotZ*)で各軸ごとの角度を設定します。
・角度はラジアンを単位とします。角度をラジアンに変換する場合、RAD関数(→p.98参照)を用います。

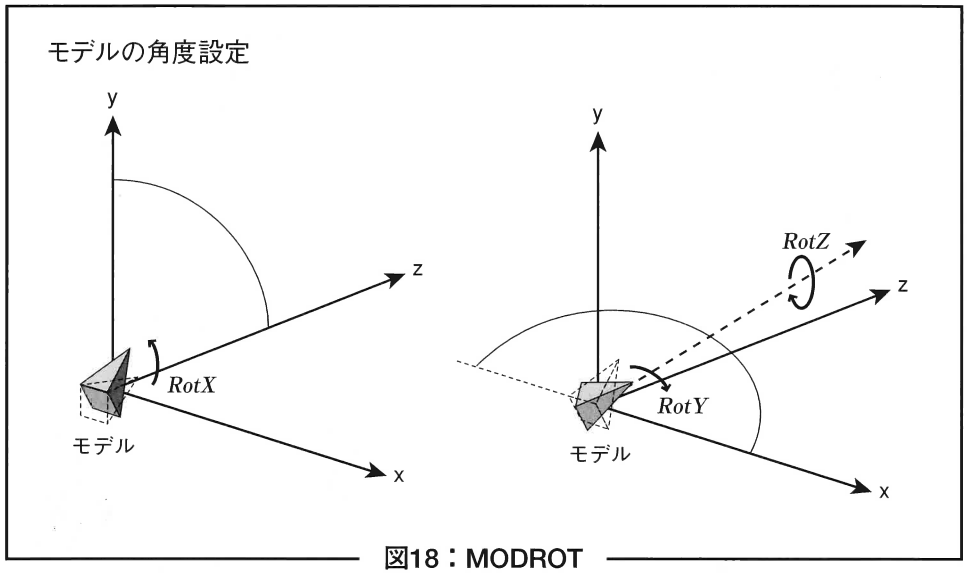


図18：MODROT

◆関連項目

MODJOIN (→p.77参照)　　MODPOS (→p79参照)

・例1

```
MODROT␣1,PI/2,0,0                    モデル番号1のモデルを、X軸に対しπ/2傾けます。
MODROT␣1,RAD(90),RAD(-30),RAD(45)    モデル番号1のモデルを、X軸に沿って90度、Y軸に沿って-30度、Z軸に沿って45度傾けます。
```

・例2

```
100 MODDISP␣1
110 DO
120  X=X+(PAD(1,4)-PAD(1,3))/80      X軸の傾きをコントローラ入力によって設定
130  Y=Y+(PAD(1,2)-PAD(1,1))/80      Y軸の傾きをコントローラ入力によって設定
140  Z=Z+(PAD(1,10)-PAD(1,9))/80     Z軸の傾きをコントローラ入力によって設定
150  MODROT␣1,RAD(X),RAD(Y),RAD(Z)  角度X,Y,Zの値をラジアンに変換してMODROT命令を実行します。
160  PRINT␣INT(X),INT(Y),INT(Z)      角度X,Y,Zの値の整数部を表示します。
170  FSYNC␣1
180 LOOP
190 END
```

■MODSET　モデル番号へのモデルの登録

◆書式

MODSET *ModelNo,ModelBuffNo,ObjNo*

項目	内容	備考
ModelNo	モデル番号	範囲：1～1000
ModelBuffNo	モデルバッファ番号	範囲：1～8
ObjNo	モデルバッファ内オブジェクト番号	範囲：1～10

◆解説

・モデル番号とモデルバッファに格納されたモデルを結び付けます。
・*ModelBuffNo*で指定したモデルバッファ内の*ObjNo*番のモデルデータを、*ModelNo*番のモデルに結び付けます。
・複数のモデル番号に同じモデルデータを使用し、別々のキャラクターとして扱うことができます。
・MODLOAD命令(→p.78参照)でモデルバッファに読み込まれていないモデルデータを指定するとエラーになります。

◆関連項目

MODLOAD (→p.78参照)　MODJOIN (→p.77参照)　MODOPT (→p.79参照)

・例

```
MODSET 3,1,1
```

モデル番号3番に、モデルバッファ1番内の1番目のオブジェクトを割り当てます。

■ZSORT　オブジェクトのZソート

◆書式

ZSORT *StartModelNo,EndModelNo*

項目	内容	備考
StartModelNo	Zソートを開始するモデル番号	範囲：1～1000
EndModelNo	Zソートを終了するモデル番号	範囲：1～1000

◆解説

・オブジェクト描画の際、スクリーンからのZ値でモデルをソートし、遠い順に描画する設定をします。
・設定を一度行うと、実際のZソートは描画時に自動的に行います。
・*StartModelNo*および*EndModelNo*に同じ番号を指定すると、Zソート範囲の設定を解除しZソートを行わなくなります。指定する番号は任意の数値です。
・半透明のオブジェクトを使用しない場合、設定する必要はありません。
・Zバッファ問題(→p.44参照)を軽減することができます。
・Zソートの設定をしていない場合、モデル番号順にモデルを描画します。

◆関連項目

ORDER (→p.38参照)

・例

```
ZSORT 100,200
```

モデル番号100番から200までのモデルはZソートされて遠い順に描画されます。

サウンド

■BGMLOAD　BGMの読み込み

◆書式

BGMLOAD ［*ディレクトリ名*/］*ファイル名*, *BGMバッファ番号*

項目	備考
BGMバッファ番号	範囲：1～4

◆解説

・BGMデータを指定した*ファイル名*から読み込んで、*BGMバッファ番号*に指定したBGMバッファに格納します。
・*ディレクトリ名*を指定すると、そのディレクトリからファイルを読み込みます。*ディレクトリ名*を省略すると、"SYSTEM"ディレクトリからファイルを読み込みます。

◆関連項目

BGMPLAY (→p.82参照)　SELOAD (→p.83参照)

・例

```
BGMLOAD "SOUND/BGM1",1
```

SOUNDディレクトリ内のファイルBGM1を読み込んで、BGMデータをBGMバッファ1番に格納します。

■BGMPLAY　BGMの再生

◆書式

BGMPLAY *BgmNo*

項目	内容	備考
BgmNo	BGMバッファ番号	範囲：1～4

◆解説

・*BgmNo*で指定したBGMバッファに格納されているBGMデータを再生します。

◆関連項目

BGMLOAD (→p.82参照)　SEPLAY (→p.84参照)　BGMSTOP (→p.83参照)

・例

```
BGMPLAY 3
```

BGMバッファ3番に格納されているBGMデータを再生します。

■BGMSTOP　BGMの停止

◆書式

BGMSTOP

◆解説

・BGMの再生を停止します。

◆関連項目

BGMPLAY (→p.82参照)

■SELOAD　効果音の読み込み

◆書式

SELOAD ［*ディレクトリ名/*］*ファイル名*

◆解説

・*ファイル名*に指定したファイルからSEデータを読み込みます。
*ディレクトリ名*を指定すると、そのディレクトリからファイルを読み込みます。
*ディレクトリ名*を省略すると、"SYSTEM"ディレクトリからファイルを読み込みます。

◆関連項目

SEPLAY (→p.84参照)

・例

```
SELOAD "SE/SE01"
```

"SE"ディレクトリ内のSEファイルSE01を読み込みます。

■SEPLAY　効果音の再生

◆書式

SEPLAY *SeNo,LeftVol,RightVol*

項目	内容	備考
SeNo	SE番号	範囲：1～128
LeftVol	左ボリューム	範囲：0～255
RightVol	右ボリューム	範囲：0～255

◆解説

・*SeNo*で指定した効果音を再生します。
・*LeftVol,RightVol*で指定した左スピーカーおよび右スピーカーのボリュームで効果音の再生をします。

◆関連項目

SELOAD (→p.84参照)

・例／効果音のテストプログラム

```
100 DO␣WHILE␣PAD(1,19)=0
110  CLEAR
120  SE=SE+(PAD(1,1)-PAD(1,2))*0.001
130  V=MAX(-255,MIN(255,V+(PAD(1,9)-PAD(1,10))*0.005))
140  L=MIN(255,255+V)
150  R=MIN(255,255-V)
160  IF␣PAD(1,6)>0␣THEN
170   IF␣FLGB=0␣THEN
180    SEPLAY␣SE,L,R
190    FLGB=1
200   END␣IF
210  ELSE
220   FLGB=0
230  END␣IF
240  SE=MAX(1,MIN(100,SE))
250  PRINT␣"SE:";INT(SE),"L:",INT(L),"R:";INT(R)
260  FSYNC␣1
270 LOOP
```

＜解説＞
120　コントローラの方向キー左・右で再生する効果音の番号を変更します。
130　コントローラのL1ボタン,R1ボタンで左右のボリュームを調節します。
140　左のボリュームの設定
150　右のボリュームの設定
160　○ボタンを押すと効果音を再生します。

■VOLUME　サウンドの音量設定

◆書式

VOLUME *Volume*

項目	内容	備考
Volume	音量	範囲：0～255

◆解説

・サウンドの音量を*Volume*で指定した値に設定します。

◆関連項目

BGMPLAY (→p.82参照)　SEPLAY (→p.84参照)

・例／ボリュームを調節するプログラム

```
100 BGMPLAY␣1
110 TRIDISP␣1
120 DO␣UNTIL␣PAD(1,19)=1
130  CLEAR
140  V=MAX(1,MIN(255,V+(PAD(1,3)-PAD(1,4))*0.005))
150  VOLUME␣V
160  TRICOL3␣1,255,V,V,128,V,255,0,128,0,255,V,128
170  TRIPOS␣1,320,220,320-V,220-V*0.8,320+V,220-V*0.8,0
180  PRINT␣"VOL:";INT(V)
190  FSYNC␣1
200 LOOP
210 INIT
```

＜解説＞

100 BGMデータを再生します。
140 コントローラの方向キー上・下でボリュームを1～255の範囲に設定します。
150 ボリュームを変数Vの値に設定します。

文字列関数

■CHR$　JISコード番号に対応する文字の取得

◆書式

CHR$(*x*)

項目	内容	備考
x	JISコード番号	0～255

◆解説

・JISコード番号に対応する文字を返します。

表1：JISコード表

code	キー	code	キー	code	キー	code	キー	code	キー
0	・	63	?	97	a	163	｣	197	ﾅ
・	・	64	@	98	b	164	、	198	ﾆ
・	・	65	A	99	c	165	・	199	ﾇ
・	・	66	B	100	d	166	ヲ	200	ﾈ
33	!	67	C	101	e	167	ァ	201	ﾉ
34	"	68	D	102	f	168	ィ	202	ﾊ
35	#	69	E	103	g	169	ゥ	203	ﾋ
36	$	70	F	104	h	170	ェ	204	ﾌ
37	%	71	G	105	i	171	ォ	205	ﾍ
38	&	72	H	106	j	172	ャ	206	ﾎ
39	'	73	I	107	k	173	ュ	207	ﾏ
40	(	74	J	108	l	174	ョ	208	ﾐ
41	)	75	K	109	m	175	ッ	209	ﾑ
42	*	76	L	110	n	176	ー	210	ﾒ
43	+	77	M	111	o	177	ｱ	211	ﾓ
44	,	78	N	112	p	178	ｲ	212	ﾔ
45	-	79	O	113	q	179	ｳ	213	ﾕ
46	.	80	P	114	r	180	ｴ	214	ﾖ
47	/	81	Q	115	s	181	ｵ	215	ﾗ
48	0	82	R	116	t	182	ｶ	216	ﾘ
49	1	83	S	117	u	183	ｷ	217	ﾙ
50	2	84	T	118	v	184	ｸ	218	ﾚ
51	3	85	U	119	w	185	ｹ	219	ﾛ
52	4	86	V	120	x	186	ｺ	220	ﾜ
53	5	87	W	121	y	187	ｻ	221	ﾝ
54	6	88	X	122	z	188	ｼ	222	ﾞ
55	7	89	Y	123	{	189	ｽ	223	ﾟ
56	8	90	Z	124	¦	190	ｾ	・	・
57	9	91	[	125	}	191	ｿ	・	・
58	:	92	¥	126	~	192	ﾀ	・	・
59	;	93	]	・	・	193	ﾁ	・	・
60	<	94	^	・	・	194	ﾂ	・	・
61	=	95	_	161	。	195	ﾃ	・	・
62	>	96	`	162	「	196	ﾄ	255	・

◆関連項目

・ORD(→p.88参照)

■DATE$　日付の文字列の取得

◆書式

DATE$

◆解説

・現在の日付を文字列として返します。
・日付の文字列は、西暦4桁、月2桁、日2桁を連続した形で得られます。

◆関連項目

TIME (→p.100参照)　TIME$ (→p.90参照)

■LCASE$　英大文字を英小文字に置換

◆書式

LCASE$(*a$*)

項目	内容	備考
a$	文字列式	

◆解説

・*a$*の値に含まれる半角英大文字をすべて半角英小文字に置き換えて、その値を返します。
・半角英大文字以外の文字は無視します。

◆関連項目

UCASE$ (→p.91参照)

・例

```
LCASE$("Basic␣Studio␣1.0")
```

文字列"basic studio 1.0"を得ます。

■LEN　文字列の長さを得る

◆書式

LEN(*a$*)

項目	内容	備考
a$	文字列式	

◆解説

・*a$*の値の長さを評価し、その値を返します。
・全角文字は1文字につき、2の長さを持つ文字として評価します。
・空白も1文字の長さとして評価します。

・例

```
PRINT␣LEN("BASICプログラム")
```

引数に与えた文字列の長さを表示します。全角文字は1文字につき2の長さを持ちます。

＜実行結果＞

```
15
```

■LTRIM$　文字列に先行する空白の削除

◆書式

LTRIM$(*a$*)

項目	内容	備考
a$	文字列式	

◆解説

・*a$*の値を評価し、空白以外の文字に先行する空白をすべて削除して得られる文字列を返します。

◆関連項目

RTRIM$ (→p.90参照)

・例

```
PRINT␣LTRIM$("   A  B cd")
```

引数の文字列内で先行する空白を削除した、文字列を表示します。

＜実行結果＞

```
A  B cd
```

■ORD　文字のJISコード番号を得る

◆書式

ORD(*a$*)

項目	内容	備考
a$	文字列式	

◆解説

・*a$*を評価し、それに対応するJISコード番号を返します。

◆関連項目

CHR$ (→p.86参照)

・例

```
PRINT␣ORD("#")
```

文字"#"のJISコード番号を表示します。

■POS　一致した文字列の先頭文字位置を得る

◆書式

POS(*a$*, *b$*)

項目	内容	備考
a$	文字列式	
b$	文字列式	

◆解説

・*a$*の値の中で、*b$*の値と最初に一致する部分の、先頭の文字の位置を返します。
・一致する部分がない場合、0を返します。
・*b$*が空文字""の場合、*a$*の値と関係なく常に1となります。

・例

```
POS("ABCDEFG","CD")
POS("ABCDEF","XYZ")
PRINT␣POS("BASIC","")
```

3文字目の文字"C"に一致し、3を返します。
一致しないため0を返します。
空文字を指定しているため、1を返します。

■REPEAT$　繰り返し連結した文字列を得る

◆書式

REPEAT$(*a$*, *x*)

項目	内容	備考
a$	繰り返す文字列式	
x	繰り返し回数	範囲：0以上

◆解説

・*a$*の値を、*x*回だけ繰り返し連結し、その文字列を返します。
・*x*は0以上を指定します。0を指定すると空文字列を返します。
・返す文字列の長さが512より大きいとエラーになります。

◆例

```
A$=REPEAT$("ABC",5)
```

文字列"ABC"を5回繰り返し連結した文字列を、文字列変数A$に代入します。

■RTRIM$　後続する空白の削除

◆書式

RTRIM$(*a$*)

項目	内容	備考
a$	文字列式	

◆解説

・*a$*の値の中で、後続する空白をすべて削除した文字列を返します。

◆関連項目

LTRIM$ (→p.88参照)

・例

```
RTRIM$("  A  B C  ")
```
後続する空白を削除し、文字列" A B C"を得ます。

■STR$　数値から文字列を得る

◆書式

STR$(*x*)

項目	内容	備考
x	数値式	

◆解説

・*x*を評価し、その値を文字列として返します。

◆関連項目

VAL (→p.91参照)

・例

```
A$=STR$(5/2)
B$=STR$(-3E2)
```
数値式5/2を評価し、結果の値2.5を文字列として文字列変数A$に代入します。
指数表記-3E2を評価し、結果の値-300を文字列として文字列変数B$に代入します。

■TIME$　現在の時刻を文字列で得る

◆書式

TIME$

◆解説

・現在の時刻を求め、文字列として返します。
・時刻は24時間表示によって表示され、時、分、秒はコロン(:)で区切られます。
・午前0時のTIME$関数が返す値は"00:00:00"です。

◆関連項目

TIME (→p.100参照)　　DATE$ (→p.87参照)

・例

```
A$=TIME$
```
現在の時刻を文字列変数A$に代入します。

■UCASE$　英小文字を英大文字に置換

◆書式

UCASE$(*a$*)

項目	内容	備考
a$	文字列式	

◆解説

・*a$*の評価値に含まれる半角英小文字をすべて半角英大文字に置き換えて、その値を返します。
・半角英小文字以外の文字は無視します。

◆関連項目

LCASE$ (→p.87参照)

・例

```
UCASE$("Basic␣Studio␣1.0")
```
文字列"BASIC STUDIO 1.0"を得ます。

■VAL　数値を表す文字列から数値を得る

◆書式

VAL(*a$*)

項目	内容	備考
a$	文字列式	

◆解説

・*a$*の評価値を数値として返します。
・*a$*には0～9の半角数字、正号、負号、小数点、指数記号を用いることができます。
・*a$*に、数値として評価できる文字以外の文字列が先行するとエラーになります。
・*a$*に、数値として評価できる文字列が複数ある場合、最初の値のみを返します。
・*a$*の前後にある空白は無視します。

◆関連項目

STR$ (→p.90参照)

・例

```
VAL("-12.34")
VAL("300"&".400")
VAL("1E3")
VAL("34POINTS")
VAL("100␣OR␣200")
VAL("    2E-3    ")
```
数値-12.34を得ます。
数値300.4を得ます。
指数表現1E3を評価した数値1000を得ます。
数値34を得ます。
数値100を得ます。
数値0.002を得ます。

数値関数

■ABS　絶対値の取得

◆書式

ABS(x)

項目	内容	備考
x	数値式	

◆解説

・xを評価し、その絶対値を返します。

■ACOS　逆余弦の取得

◆書式

ACOS(x)

項目	内容	備考
x	数値式	単位：ラジアン

◆解説

・xを評価し、その逆余弦を返します。
・返り値の範囲：0≦ACOS(x)≦π

■ANGLE　角度の取得

◆書式

ANGLE(x,y)

項目	内容	備考
x	X軸ベクトル	
y	Y軸ベクトル	

◆解説

・原点と座標(x,y)を結ぶベクトルの、X軸に対する角度を返します。
・返り値はベクトルを単位とします。範囲は-π≦ANGLE(x,y)≦πの間です。
・反時計回りを正とします。
・引数x,yともに0を指定するとエラーになります。

・例

```
ANGLE(1,1)
```

値0.785398(=45°)を得ます。

■ASIN　逆正弦の取得

◆書式

ASIN(*x*)

項目	内容	備考
x	数値式	

◆解説

・*x*を評価し、その逆正弦を返します。
・返り値はラジアンを単位とします。範囲は-π/2≦ASIN(*x*)≦π/2の間です。

■ATN　逆正接の取得

◆書式

ATN(*x*)

項目	内容	備考
x	数値式	

◆解説

・*x*を評価し、その逆正接を返します。
・返り値はラジアンを単位とします。範囲は-π/2＜ATN(*x*)＜π/2の間です。

■COS　余弦の取得

◆書式

COS(*x*)

項目	内容	備考
x	角度	単位：ラジアン

◆解説

・角度*x*の余弦を返します。

■DEG　度を単位とする角度の取得

◆書式

DEG(*x*)

項目	内容	備考
x	角度	単位：ラジアン

◆解説

・ラジアンを単位とする角度を、度を単位とする数値に変換して返します。

・例

```
DEG(PI)
```
値180を得ます。

■EXP　自然対数の底eの乗数の取得

◆書式

EXP(*x*)

項目	内容	備考
x	乗数	

◆解説

・指数関数、すなわち自然対数の底eの*x*乗を返します。

■FP　小数部の取得

◆書式

FP(*x*)

項目	内容	備考
x	数値式	

◆解説

・数値式*x*を評価し、その少数部を返します。

■INT　引数の値を超えない最大の整数の取得

◆書式

INT(*x*)

項目	内容	備考
x	数値式	

◆解説

・xを評価し、その値を超えない最大の整数を返します。

・例

```
INT(1.3)　　値1を得ます。
INT(-1.3)　　値-2を得ます。
```

■IP　整数部の取得

◆書式

IP(*x*)

項目	内容	備考
x	数値式	

◆解説

・数値式*x*を評価し、その整数部を返します。

・例

```
IP(1.3)　　値1を得ます。
IP(-1.3)　　値-1を得ます。
```

■LOG　自然対数の取得

◆書式

LOG(*x*)

項目	内容	備考
x	数値式	

◆解説

・*x*を評価し、その自然対数を返します。
・xの評価値は正でなければなりません。

■LOG10　常用対数の取得

◆書式

LOG10(*x*)

項目	内容	備考
x	数値式	

◆解説

・*x*を評価し、その常用対数を返します。
・*x*の評価値は正でなければなりません。

■LOG2　2を底とする対数の取得

◆書式

LOG2(*x*)

項目	内容	備考
x	数値式	

◆解説

・*x*を評価し、2を底とするその評価値の対数を返します。
・*x*の評価値は正でなければなりません。

■MAX　大きい方の値の取得

◆書式

MAX(*x*,*y*)

項目	内容	備考
x	数値式	
y	数値式	

◆解説

・*x*および*y*を評価し、大きい方の値を返します。

・例

```
100 X=MAX(0,X)
110 X=MIN(640,X)
```

変数Xの値が0より大きい場合変数Xを、それ以外の場合0を代入します。
変数Xの値が640より小さい場合変数Xを、それ以外の場合640を代入します。

■MIN　小さい方の値の取得

◆書式

MIN(*x*,*y*)

項目	内容	備考
x	数値式	
y	数値式	

◆解説

・*x*および*y*を評価し、小さい方の値を返します。

・例

```
X=MIN(640,MAX(0,X))
```

変数Xに MAX(0,X)の結果と640を比較し小さい方の値を代入します。

■MOD　yを法とするxの値の取得

◆書式

MOD(*x*,*y*)

項目	内容	備考
x	数値式	
y	数値式	

◆解説

・*x*および*y*を評価し、*y*を法とする*x*の値を返します。
・*y*の評価値を0にするとエラーになります。

■PI　円周率πの取得

◆書式

PI

◆解説

・円周率π(=3.14159)を得ます。

■RAD　ラジアンを単位とする角度の取得

◆書式

RAD(*x*)

項目	内容	備考
x	数値式	単位：度

◆解説

・度を単位とする角度を、ラジアンを単位とする数値に変換して返します。

■REMAINDER　除算による余りの取得

◆書式

REMAINDER(*x*,*y*)

項目	内容	備考
x	割られる数	
y	割る数	

◆解説

・*x*を*y*で割った余りを返します。
・*y*を0にするとエラーになります。

■RND　乱数の取得

◆書式

RND

◆解説

・0以上1未満の乱数を返します。

◆関連項目

RANDOMIZE(→p.20参照)

・例

```
INT(RND*100)
```

0～99 のいずれかの数値を得ます。

■SGN　符号の取得

◆書式

SGN(*x*)

項目	内容	備考
x	数値式	

◆解説

・*x*を評価し、その符号を返します。
・*x*が負であれば-1、*x*が正であれば1、*x*が0であれば0を返します。

■SIN　正弦の取得

◆書式

SIN(*x*)

項目	内容	備考
x	角度	単位：ラジアン

◆解説

・角度xの正弦を返します。

■SQR　平方根の取得

◆書式

SQR(*x*)

項目	内容	備考
x	数値式	

◆解説

・*x*を評価し、その平方根を返します。
・*x*を負数にするとエラーになります。

■TAN　正接の取得

◆書式

TAN(*x*)

項目	内容	備考
x	角度	単位：ラジアン

◆解説

・角度*x*の正接を返します。

■TIME　午前0時からの経過秒数の取得

◆書式

TIME

◆解説

・その日の午前0時からの経過秒数を返します。

■COLLISION　モデル同士の当たり判定

◆書式

COLLISION(*m*,*s*,*e*)

項目	内容	備考
m	当たり判定するモデル番号	
s	判定開始モデル番号	
e	判定終了モデル番号	

◆解説

・*m*番のモデルに対し、モデル番号*s*～*e*までのモデルと当たり判定を行い、衝突している最初のモデル番号を返します。
・どのモデルとも衝突していない場合、0を返します。
・モデル番号*s*～*e*の範囲内にモデル番号*m*が含まれている場合、無視します。
・*s*>*e*の場合、エラーになります。
・モデルデータ内に当たり判定用の設定をしておく必要があります。

◆関連項目

GETCOLL(→p.70参照)

・例

```
N=COLLISION(7,5,10)
```

モデル番号7番のモデルに対し、5～10番のモデルとの当たり判定を行い、最初に接触したモデルの番号を変数Nに代入します。モデル番号7番自身との当たり判定は行いません。

■INKEY$　キーボードからの入力文字の取得

◆書式

INKEY$

◆解説

・現在押されているキーボード上のキーの文字を返します。
※"PlayStation 2"（本体）のUSB端子に、USBキーボードが接続されている場合のみ有効です。

◆関連項目

INPUT (→p.17参照)　KEYCODE1 (→p.102参照)　KEYCODE2 (→p.103参照)

・例

```
A$=INKEY$
```

現在押されているキーボード上のキーの文字を文字列変数A$に代入します。

■KEYCODE1　キーボードから入力されたキーのコード取得

◆書式

KEYCODE1

◆解説

・現在押されているキーのキーコードを返します。
・キーが押されていない場合、0を返します。
※"PlayStation 2"（本体）のUSB端子に、USBキーボードが接続されている場合のみ有効です。

表2：キーコード表

code	キー	code	キー	code	キー	code	キー
0	（キーが押されていない）	28	Y	56	/	84	/（テンキー）
1	（エラーコード）	29	Z	57	CapsLock	85	*（テンキー）
2		30	1	58	F1	86	-（テンキー）
3		31	2	59	F2	87	+（テンキー）
4	A	32	3	60	F3	88	Enter（テンキー）
5	B	33	4	61	F4	89	1（テンキー）
6	C	34	5	62	F5	90	2（テンキー）
7	D	35	6	63	F6	91	3（テンキー）
8	E	36	7	64	F7	92	4（テンキー）
9	F	37	8	65	F8	93	5（テンキー）
10	G	38	9	66	F9	94	6（テンキー）
11	H	39	0	67	F10	95	7（テンキー）
12	I	40	ENTER	68	F11	96	8（テンキー）
13	J	41	ESC	69	F12	97	9（テンキー）
14	K	42	Backspace	70	PrintScreen	98	0（テンキー）
15	L	43	TAB	71	ScrollLock	99	.（テンキー）
16	M	44	スペース	72	Break	・	・
17	N	45	ー	73	Insert	・	・
18	O	46	^	74	Home	・	・
19	P	47	@	75	PageUp	・	・
20	Q	48	[	76	Delete	・	・
21	R	49		77	End	・	・
22	S	50	]	78	PageDown	・	・
23	T	51	;	79	→	135	＼
24	U	52	:	80	←	136	カタカナひらがな
25	V	53	半角/全角	81	↓	137	￥
26	W	54	,	82	↑	138	変換
27	X	55	.	83	NumLock	139	無変換

◆関連項目

KEYCODE2 (→p.103参照)　INKEY$ (→p.101参照)　INPUT (→p.17参照)

・例

```
100 DO
110  PRINT␣KEYCODE1
120 LOOP
```

現在押下されているキーのキーコードを取得し、表示します。

■KEYCODE2　キーボードから入力された特殊キーのコード取得

◆書式

KEYCODE2

◆解説

・現在押されているshiftキーやctrlキーなどの特殊キーのキーコードを返します。
・なにも押されていない場合、0を返します。
・同時に押されている特殊キーのキーコードの合計値を返します。
※"PlayStation 2"（本体）のUSB端子に、USBキーボードが接続されている場合のみ有効です。

◆関連項目

INKEY$ (→p.101参照)　INPUT (→p.17参照)　KEYCODE1 (→p.102参照)

・例

```
100 DO
110  PRINT␣KEYCODE2                押されている特殊キーのコード合計値を表示します
120 LOOP
130 END
```

■OBJPOLY　モデルのポリゴン数の取得

◆書式

OBJPOLY(*x*,*y*)

項目	内容	備考
x	モデルバッファ番号	
y	オブジェクト番号	

◆解説

・モデルバッファ*x*番内の、*y*番目のオブジェクトの総ポリゴン数を返します。

・例

```
P=OBJPOLY(1,2)
```

モデルバッファ1番内の2番目のオブジェクトの総ポリゴン数を、変数Pに代入します。

■PAD　コントローラの入力情報の取得

◆書式

PAD(*x*,*y*)

項目	内容	備考
x	コントローラ端子番号	
y	ボタン番号	

◆解説

・コントローラのボタンの入力値を取得します。
・*x*で指定した番号のコントローラ端子に接続されたコントローラの入力情報を取得します。

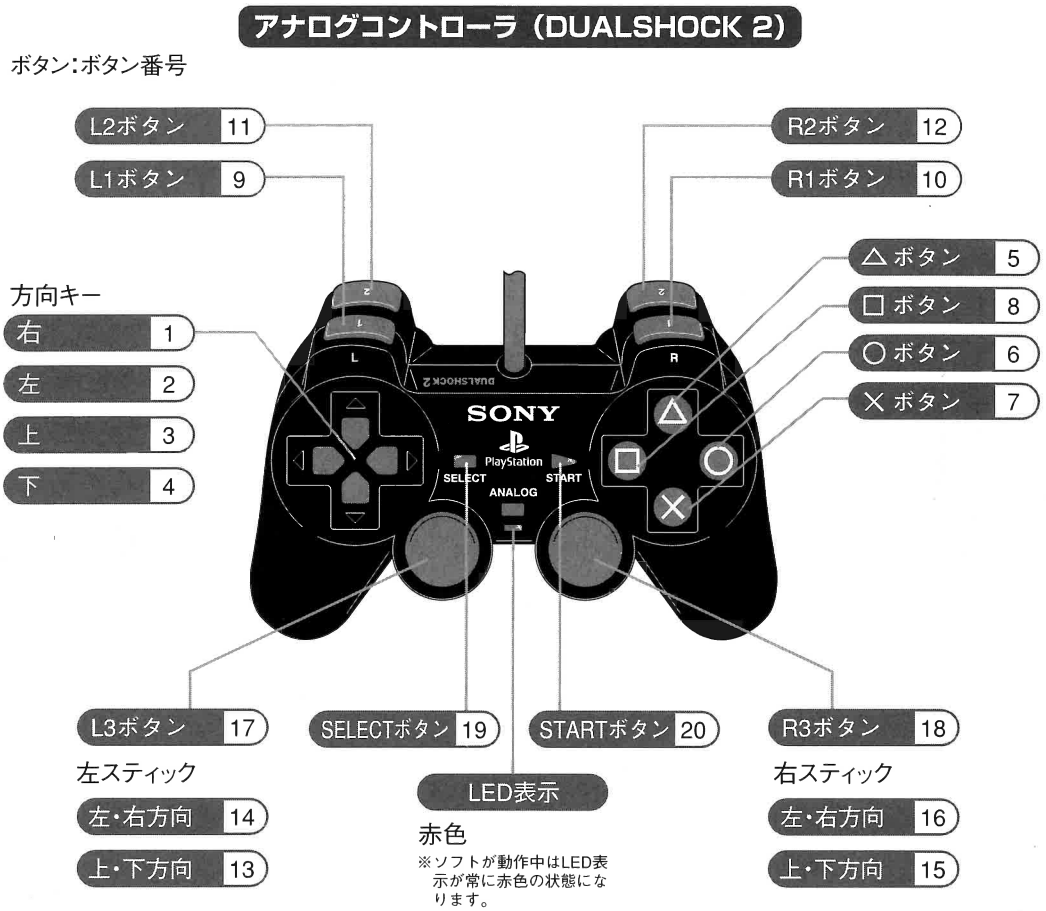


図19：コントローラのボタン番号

表3：コントローラのボタン対応

ボタン番号	ボタン	範　囲	静止状態
1	方向キー右	0-255	0
2	方向キー左	0-255	0
3	方向キー上	0-255	0
4	方向キー下	0-255	0
5	△ボタン	0-255	0
6	○ボタン	0-255	0
7	×ボタン	0-255	0
8	□ボタン	0-255	0
9	L1ボタン	0-255	0
10	R1ボタン	0-255	0

ボタン番号	ボタン	範　囲	静止状態
11	L2ボタン	0-255	0
12	R2ボタン	0-255	0
13	左スティック（上・下方向）	0-255	127付近
14	左スティック（左・右方向）	0-255	127付近
15	右スティック（上・下方向）	0-255	127付近
16	右スティック（左・右方向）	0-255	127付近
17	L3ボタン	0-1	0
18	R3ボタン	0-1	0
19	SELECTボタン	0-1	0
20	STARTボタン	0-1	0

・例

```
100 TEXTMODE␣2
110 FOR␣A=1␣TO␣16
120  TILEDISP␣A
130 NEXT␣A
140 DO
150  CLEAR
160  PRINT␣"  右␣左␣上␣下␣△␣○␣×␣□␣L1␣R1␣L2␣R2␣L3Y,X␣R3Y,X"
170  FOR␣B=1␣TO␣16                                    ボタン番号1～16のボタンについて入力量を調べます。
180   D=PAD(1,B)                                      ボタン入力量を変数Dに代入します。
190   TILECOL␣B,␣D,D,B*15,␣128                        ボタンの入力量をもとに色を設定します。
200   TILEPOS␣B,B*36+5,20,B*36+28,20+D/2,␣0           ボタンの入力量をもとにタイルプリミティブの位置を設定します。
210  NEXT␣B
220  FSYNC␣1
230 LOOP
240 END
```

■SCRCHR$　画面上の文字の取得

◆書式

SCRCHR$(*x*, *y*)

項目	内容	備考
x	テキストX座標値	範囲：0～37(テキストモード1)、0～49(テキストモード2)
y	テキストY座標値	範囲：0～22

◆解説

・テキスト座標(*x*,*y*)で指定した位置に表示されている文字を取得します。
・テキスト座標の範囲外の値を指定するとエラーになります。
指定した位置に全角文字がある場合、取得される値は不正になります。

・例

```
SCRCHR$(34,12)          テキスト座標(34,12)にある文字を得ます。
```

索引

用語

演算子 4
数値 5
文字列 5
配列 6
副プログラム 7
グラフィックス仕様 32
色指定 32
スクリーンについて 43
Zバッファ方式 44
Zバッファ問題 44
Zソート 45
3次元空間の表現について 61
モデル 62
カメラ 63
光源 64
フォグ 64

記号

＋ 4
－ 4
＊ 4
／ 4
＾ 4
e(指数記号) 4
＝ 4
＞ 4
＜ 4
>=,=> 4
<=,=< 4
<>,>< 4
AND 4
OR 4
NOT 4

命令

ABS 92
ACOS 92
ALTEST 65
AMBCOL 65
ANGLE 92
ASIN 93
ATN 93
BGCOL 46
BGMLOAD 82
BGMPLAY 82
BGMSTOP 83
BSCOL 46
CALL 9
CAMLENS 66
CAMPOS 66
CAMROT 67
CAMVEC 68
CHR$ 86
CLEAR 33
COLLISION 101
CONTINUE 25
COS 93
DATE$ 87
DEG 94
DELETE 25
DEVICE 33
DIM 11
DO-LOOP 12
EFFECT 47
END 13
EXP 94
FILES 26
FKEY 26
FOGCOL 69
FOGSET 69
FOR TO STEP 14
FP 94
FSYNC 34
GETCOLL 70
GETLPOS 71
GETSPOS 72
GETWPOS 73
GOTO 15
IF THEN-END IF 16
INIT 34
INKEY$ 101
INPUT 17
INT 95
IP 95
KEYCODE1 102
KEYCODE2 103
KILL 26
LABEL 35
LCASE$ 87
LEN 87
LET 18
LIGCOL 74
LIGVEC 75

LINECOL 47
LINEDISP 48
LINEOFF 48
LINEPOS 49
LIST 27
LOAD 28
LOCATE 35
LOG 95
LOG10 96
LOG2 96
LTRIM$ 88
MAX 96
MEMORY 28
MIN 97
MOD 97
MODDISP 76
MODJOIN 77
MODLOAD 78
MODOFF 78
MODOPT 79
MODPOS 79
MODROT 80
MODSET 81
MOUAREA 36
MOUCOL 36
MOUDISP 36
MOUGET 37
MOUOFF 37
MOUPAT 37
MOUSET 38
NEW 28
OBJPOLY 103
ON-GOTO 19
OPTION BASE 19
ORD 88
ORDER 38
PAD 104
PALCOPY 49
PALLOAD 50
PI 97
POS 89
PRINT 20
QUIT 28
RAD 98
RANDOMIZE 20
READ/DATA 21
REM 22
REMAINDER 98
RENUMBER 29
REPEAT$ 89
RESTORE 22
RND 98
RTRIM$ 90
RUN 30
SAVE 30
SCRCHR$ 105
SCREEN 39
SELECT CASE 23
SELOAD 83
SEPLAY 84
SGN 99
SIN 99
SPRDISP 50
SPROFF 50
SPRPAL 51
SPRPOS 52
SPRPOS2 53
SPRTEX 54
SQR 99
STOP 24
STR$ 90
SUB-END SUB 10
TAN 100
TEXCOPY 55
TEXLOAD 55
TEXTCOL 40
TEXTMODE 40
TILECOL 56
TILEDISP 56
TILEOFF 57
TILEPOS 57
TIME 100
TIME$ 90
TRICOL 58
TRICOL3 59
TRIDISP 59
TRIOFF 60
TRIPOS 60
UCASE$ 91
VAL 91
VARLOAD 41
VARSAVE 41
VIBRATE 42
VIEW 31
VOLUME 85
ZSORT 81

BASICSTUDIO

パワフルゲーム工房

リファレンスマニュアル

株式会社アートディンク
〒261-8701 千葉市美浜区中瀬2-6 WBGマリブウエスト22階
URL http://www.artdink.co.jp/

©2001 ARTDINK.All Rights Reserved.
"PlayStation"および"DUALSHOCK"は株式会社ソニー・コンピュータエンタテインメントの登録商標です。
ソニー製"プレイステーション 2"対応プリンター popegg 対応。
『popegg』はソニー株式会社の商標です。
その他表記されている商品名、社名は、一般に各社の商標または登録商標です。
2001年4月5日 初版発行